The Motor Industry Software Reliability Association

MISRA c/o Electrical Group, MIRA, Watling Street, Nuneaton, Warwickshire, CV10 0TU, UK.
Telephone: (024) 7635 5290. Fax: (024) 7635 5070. E-mail: misra@mira.co.uk Internet: http://www.misra.org.uk

Report 2

Integrity

February 1995

PDF version 1.0, January 2001

Acknowledgements

The following were contributors to this report:

Frank O'Neill	Lucas Electronics
Nick Bennett	Lucas Electronics
David Newman	Ford Motor Company Ltd
Ian Kendall	Jaguar Cars Ltd
Roger Rivett	Rover Group Ltd
Paul Edwards	Rover Group Ltd
Chris Marshall	AB Automotive Electronics Ltd
Paul Manford	AB Automotive Electronics Ltd
Peter Jesty	University of Leeds
Keith Hobley	University of Leeds
Neil Andrewartha	Rolls Royce and Associates Ltd
Vivien Hamilton	Rolls Royce and Associates Ltd
John Fox	The Centre for Software Engineering Ltd
Brian Whitfield	The Centre for Software Engineering Ltd

Summary

In order to ensure the safe operation of safety-related programmable systems, it is necessary to recognise the various possible causes of failures, and to ensure that adequate precautions are taken against each one. As part of a safety analysis controllability categories are assigned to each hazard; this defines the Integrity Level necessary for the design of the system associated with that hazard. This document is concerned with the software that may be a part of a system, though when necessary more general system and managerial aspects are discussed as well. It gives guidance on how the different levels of confidence can be achieved through the use of appropriate design, development and quality management processes during the system life-cycle. It also recommends that some form of independent assessment should be performed to confirm that the necessary degree of confidence has indeed been obtained, in particular for the higher Integrity Levels. The overall aim is to provide a degree of confidence in the final system, which corresponds to the Integrity Level, that the risks associated with it are indeed at an acceptable level.

Recommendations

I Introduction

An ECU in a motor vehicle is required to have a high level of integrity because there are requirements that it should not cause:

- harm to humans;
- legislation to be broken;
- undue traffic disruption;
- damage to property or the environment;
- undue financial loss to either the manufacturer or the owner.

By achieving the required level of integrity, the risks associated with each of the requirements listed above should be reduced to an acceptable level. Higher levels of integrity are achieved by the increased use of specialised methods and design requirements. This leads to an increasing cost, both of the development and of the end product.

After assigning an Integrity Level to the system the safety classification procedure provides guidance as to what development techniques and procedures should be used when designing the system, and the degree of rigour with which they should be applied. This ensures that the measures taken are necessary and sufficient for the system being designed.
The use of Integrity Levels permits a reasoned argument for assessing the degree of confidence that one should have in a system, which is commensurate with the inherent risk associated with the system function.

The recommendations and processes described in this document seek to enable a developer firstly to assess a system to determine its Integrity Level and secondly to adopt a suitable development process in order to achieve the confidence level required in the software. The recommendations can also be used by assessors and procurers of such systems to help them determine whether the methods adopted by the system developers are suitable for the proposed application.

I.i Recommendations

1. A preliminary hazard analysis should be performed early in the life-cycle to determine the initial Integrity Level and the high-level safety requirements. The reasons for discounting potential hazards which are the result of extremely unlikely scenarios should be recorded.

2. Once identified, each hazard should be assessed to determine its controllability category. The controllability approach was developed as part of the project DRIVE Safely[6].

3. The hazard with the highest controllability category will determine the Integrity Level

of the system. This hazard analysis should be followed down the hierarchy to each system sub-component (including software) to determine the Integrity Level required for each component.

4. The system design may reduce the Integrity Level required for particular sub-systems. For example a failure mode set to a safe state by a hardware interlock of a suitable Integrity Level would remove that potential failure from the software hazard list. The hazard analysis therefore needs to be maintained throughout the development.

5. Once an Integrity Level for the software has been determined, an appropriate development approach must be defined in order to gain the required confidence in the software.

6. An assessment process should be performed to demonstrate that the risks associated with the final system are at an acceptable level.

7. The higher levels of integrity should require greater independence of assessment in order to ensure that there is no bias from the development team, nor misplaced pressure from management.

8. Testing on its own is not adequate for assessing safety-related software. The degree of confidence in a given piece of software depends on the quality and the extent of the available information. This should cover the specification, design, implementation and testing of the software.

9. The higher levels of integrity require more information and more rigorous application of software engineering techniques.

10. The development approach adopted should be documented (for example by producing a Software Development Plan, Quality Plan, including a Safety Plan, and Configuration Plan). This design material should define:

    a) project organisation.
    b) procedures to be followed.
    c) management processes to ensure compliance with the procedures in (b).
    d) planned management reviews of the effectiveness of the procedures in (b) and processes in (c).
    e) defined documents together with their authors and readers.

II Safety analysis

A preliminary safety analysis should be carried out as part of the requirements analysis phase and a more detailed one carried out as the design progresses. Safety analysis is highly iterative and should be considered as a continuous process during requirements and design.

The safety analysis is based on a hazard analysis using categories of controllability which are defined as follows:

Uncontrollable

This relates to failures whose effects are not controllable by the vehicle occupant(s), and which are most likely to lead to extremely severe outcomes. The outcome cannot be influenced by a human response.

Difficult to Control

This relates to failures whose effects are not normally controllable by the vehicle occupant(s) but could, under favourable circumstances, be influenced by a mature human response. They are likely to lead to very severe outcomes.

Debilitating

This relates to failures whose effects are usually controllable by a sensible human response and, whilst there is a reduction in the safety margin, can usually be expected to lead to outcomes which are at worst severe.

Distracting

This relates to failures which produce operational limitations, but a normal human response will limit the outcome to no worse than minor.

Nuisance Only

This relates to failures where safety is not normally considered to be affected, and where customer satisfaction is the main consideration.

Controllability applies to the ability of the vehicle occupants, not necessarily the driver, to control the situation following a failure (e.g. windows and seats).

The main steps involved in the determination of the initial Integrity Level are as follows:

a) List all hazards which result from all the failures of the system.
b) Assess each failure mode identified in step (a) to determine its controllability category.
c) The failure mode with the highest associated controllability category determines the Integrity Level of the system (see Table A).

Controllability Category	Acceptable Failure Rate	Integrity Level
Uncontrollable	Extremely Improbable	4
Difficult to Control	Very Remote	3
Debilitating	Remote	2
Distracting	Unlikely	1
Nuisance Only	Reasonably Possible	0

Table A — Integrity Levels

Where the Integrity Level of the system relies upon the software, this should be identified specifically in the documentation.

II.i Human factors in safety analysis

The driver and a vehicle can interact in many different ways, and these interactions should be considered carefully during safety assessment.

The following aspects may need to be considered when assessing hazards associated with the human interaction with a system:

- human reaction times
- ease of recognition of a situation
- attentiveness
- driving experience
- risk compensation (improved safety can lead to riskier behaviour)
- subversion or over-riding of system functions
- smooth and readily perceived transfer of control from system to driver
- workload of driver, especially at the moment of transfer.

It may also be necessary to consider the effects of the range of capabilities relating to:

- vision and hearing
- mental state (for example, lack of sleep, jet lag)
- physical disability.

Table B may be useful in assessing the significance of human error when interacting with a vehicle system [19].

Type of Human Behaviour	Human Error Probability
Extraordinary errors: those for which it is difficult to conceive how they could occur: stress free, with powerful cues pointing to success.	10^{-5}
Errors in regularly performed, commonplace simple tasks with minimum stress.	10^{-4}
Errors of commission such as pressing the wrong button or reading the wrong display. Reasonably complex tasks, little time available, some cues necessary.	10^{-3}
Errors of omission where dependence is placed on situation and memory. Complex, unfamiliar task with little feedback and some distraction.	10^{-2}
Highly complex task, considerable stress, little time available.	10^{-1}
Process involving creative thinking, unfamiliar, complex operation where time is short, stress is high.	10^{-1} - 1

Table B — Human Error Probability

II.ii Development approach

1. The process of system design involves partitioning the system into sub-systems, and assigning functionality to the various elements (hardware, software, etc.). The hazard analysis described above should be followed down to the components, in order to determine the Integrity Level of each component. In this way, it is possible to reduce the Integrity Level required for certain elements of the system. The Integrity Level of the software can be reduced by the use of hardware measures. These can be used to manage the more hazardous failure modes, thus removing these hazards from the software.

2. Once the Integrity Level of the software has been determined, an appropriate development approach should be defined. Table C gives guidance of the requirements for each Integrity Level.

The following notes should be read in conjunction with Table C.

a) Specification & Design

A formal method is a description, based on a self-consistent mathematical theory of axioms and rules of inference, which is amenable to objective analysis, manipulation and proof. Formal methods involve the use of formal logic for the specification and verification of software.

Automatic code generation is recommended at level 4 in order to remove

human error from the process. It is recognised that this is not possible with current technology, and that validated tools will be required to make the approach usable.

b) Languages & Compilers

Most, if not all, languages do not have precisely defined semantics. This means that not only may compilers contain faults, but different compilers for the same language may implement a given feature in different ways. In addition, some programming "features" such as pointers or recursion can cause unpredictable behaviour. This has led to the recommendation that restricted subsets are used at levels 2 and 3 and that certified compilers with proven formal semantics (not currently available) are used at level 4.

Also, since there are currently no formally proven compilers, it may be necessary to show that the machine code (object) does indeed reflect the high level language version (source) of the program. For this reason, or for reasons of required speed of execution, restricted memory availability, etc. assembly languages are still being used at all levels of integrity.

c) Configuration Management:- Products

"Products" in this context means all documents generated or used during the development process (i.e. the set of information used for assessment). Additionally for level 2 and above, the tools used should be configured.

Development Process	Integrity Level				
	0	**1**	**2**	**3**	**4**
Specification & Design	ISO 9001	Structured method.	Structured method supported by CASE tool.	Formal specification for those functions at this level.	Formal specification of complete system. Automated code generation (when available).
Languages & Compilers		Standardised structured language.	A restricted subset of a standardised structured language. Validated or tested compilers (if available).	As for 2.	Independently certified compilers with proven formal syntax and semantics (when available).
Configuration Management:- Products		All software products. Source code.	Relationships between all software products. All tools.	As for 2.	As for 2.
Configuration Management:- Processes		Unique identification. Product matches documentation. Access control. Authorised changes.	Control and audit changes. Confirmation process.	Automated change and build control. Automated confirmation process.	As for 3.
Testing		Show fitness for purpose. Test all safety requirements. Repeatable test plan.	Black box testing.	White box module testing - defined coverage. Stress testing against livelock & deadlock. Syntactic static analysis.	100% white box module testing. 100% requirements testing. 100% integration testing. Semantic static analysis.
Verification and Validation		Show tests: are suitable; have been performed; are acceptable; exercise safety features. Traceable correction.	Structured program review. Show no new faults after corrections.	Automated static analysis. Proof (argument) of safety properties. Analysis for lack of livelock and deadlock. Justify test coverage. Show tests have been suitable.	All tools to be formally validated (when available). Proof (argument) of code against specification. Proof (argument) for lack of livelock and deadlock. Show object code reflects source code.
Access for assessment		Requirements & acceptance criteria. QA & product plans. Training policy. System test results.	Design documents. Software test results. Training structure.	Techniques, processes, tools. Witness testing. Adequate training. Code.	Full access to all stages and processes.

Table C — Summary of Requirements (see Appendix G for details)

d) Configuration Management:- Process

Confirmation process at level 2 implies a means of confirming that the software has been built from identifiable components. At levels 3 and 4, the automated confirmation process is to confirm that only intended changes are made, and to perform automated impact analysis of proposed changes.

e) Testing

Test coverage at level 4 (100%) implies an assessment of the coverage against defined criteria. This should be defined in the project planning documentation. A coverage analysis approach such as linear code sequence and jumps (LCSAJ) should be used.

f) Verification and Validation

All tools which could effect the integrity of the product should be formally validated. It is not necessary to formally validate tools such as word-processors and project planning tools, and arguments may be made for not formally validating other tools (e.g. CASE tools) where the output is fully validated. Care should be taken to ensure that in-house utilities and scaffold code linking tools together are appropriately validated.

g) Access for Assessment

The degree of independence should be justified in the planning documentation based on the level of integrity and organisational issues.

Contents

Abbreviations

ABS	Anti-lock Braking System
ACEA	Association des Constructeurs Européens d'Automobiles
ALARP	As Low as Reasonably Possible
BSI	British Standards Institute
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
ECU	Electronic Control Unit
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Mode, Effects and Criticality Analysis
FTA	Fault Tree Analysis
HGV	Heavy Goods Vehicle
IEC	International Electrotechnical Commission
ITSEC	Information Technology Security Evaluation Criteria
LCSAJ	Linear Code Sequence and Jumps
LRC	Longitudinal Redundancy Check
MISRA	Motor Industry Software Reliability Association
PSV	Public Service Vehicle
RAM	Random Access Memory (Read/Write)
ROM	Read Only Memory
TCF	Technical Construction File
UK	United Kingdom

1. Introduction

The automotive industry has been seeking ways of determining an appropriate approach to safety classification for software based systems, and the degree of rigour required to provide confidence in such safety related systems. This document describes an approach to hazard assessment centred around the concept of "integrity" in order to deal with these issues. The use of Integrity Levels permits a reasoned argument for assessing the degree of confidence that one should have in a system, which is commensurate with the level of risk acceptable within that system. However the means of achieving this is not yet a fully understood and accepted process.

This document discusses possible approaches to the concept of assessing hazards using Integrity Levels, and recommends a practical solution which is relevant to the automotive industry. Later sections are concerned with the software that may be a part of a system, though when necessary more general system and managerial aspects are discussed as well. They give guidance on how the different levels of confidence can be achieved through the use of appropriate design, implementation and quality management processes during the system life-cycle. They also recommend that some form of independent assessment should also be performed to confirm that the necessary degree of confidence has indeed been obtained, in particular for the higher Integrity Levels.

The assessment of an acceptable level of risk associated with a vehicle may be complicated by the fact that it is universally accepted that road transportation involves a certain amount of risk. Indeed most drivers select a level of risk that is (much) higher than it needs to be, e.g. few drivers choose to buy cars on the basis of safety features, and many drive too fast in adverse conditions, whilst others neglect essential checks and maintenance.

Equipment failures account for only a relatively small fraction of accidents, and those that do occur in electrical and/or mechanical systems are usually well understood by the average vehicle user, and even to some degree accepted by him or her, provided the failures are not too frequent. However, as soon as some form of computer is introduced the resulting system is expected to be "perfect". This expectation, when combined with environmental issues and other legislation, may lead a manufacturer to increase the Integrity Level of a system to something higher than pure safety considerations would dictate.

2. Scope

This document, which supports the Motor Industry Software Reliability Association (MISRA) guidelines, is intended to have a five year lifetime and as such only considers road vehicles of the type in production now. Future concepts such as road trains are acknowledged but not considered in detail.

The references made to driving practice throughout this document are based on UK experience. Whilst they should also be applicable to other countries, this should be validated

before applying the recommendations elsewhere.

3. General approach

In order to assess the hazards created by using Electronic Control Units (ECU) in vehicles a framework must be developed to provide the basis for general reasoning on various aspects of the hazards. The framework used throughout various industries is based on the concept of "integrity".

A particular hazard can be said to have been assessed when it has been accurately identified and the level of integrity to which the system should be built has been determined, such that the risk (the product of the severity of the effect and its probability of occurrence) of that hazard occurring is acceptable.

The following sequence of events should be performed:

i) accurately define each hazard;
ii) define the sequence of events which can cause each hazard;
iii) define the consequences of each hazard;
iv) determine the risk associated with the occurrence of each hazard;
v) determine the Integrity Level to which the system should be built, taking into account the risks associated with each hazard;
vi) choose a plan for development which is appropriate for the Integrity Level;
vii) develop the system
viii) ensure that the level of confidence in the final system corresponds to the original Integrity Level.

The following sections analyse the concept of integrity and show ways of determining the integrity to which a system should be built. The different methods which may be used in order to achieve a given Integrity Level are then described.

4. The concept of integrity

4.1 Integrity and risk

An ECU in a motor vehicle should have integrity because there are requirements that it should not cause:

- harm to humans (safety);
- legislation to be broken;
- undue traffic disruption;
- damage to property;
- undue financial loss to either the manufacturer or the owner;

- undue impact on the environment.

By achieving the required level of integrity, the Risks associated with each of the requirements listed above should be reduced to an acceptable level.

However, the increasingly onerous application of specialised methods and design techniques leads to an increase in cost, both of development and of the end product. Thus in order to provide guidance as to how far such methods are applied, it is necessary first to determine the Integrity Level required of the ECU.

Integrity is defined in the dictionary [1] as "freedom from impairment" or "the quality of being unimpaired". The integrity of a system is therefore the degree to which that system is free from impairments.

In order to be described as "unimpaired" a system must be able "to perform its required functions, in the desired manner, under all relevant conditions, and on the occasions when it is required to so perform" [2]. Such system requirements must therefore be specified, preferably early in the system design life-cycle when assessing end user requirements. Therefore any factor which affects these requirements in any way can be considered to be a contributory factor to integrity, or the lack of integrity, of the system. Other requirements such as those due to legislation and safety must also be accounted for.

There are a variety of factors which contribute towards the integrity of the system, and the requirements for being unimpaired include the absence of incorrect or ill timed actions, as well as the presence of correct actions. With complex systems, and particularly where software is involved, there are considerable difficulties in determining a precise, quantified, value for the final integrity achieved.

The integrity required of an ECU depends on the risk associated with the ECU and its application [3]. Risk is a property of the Undesired Events associated with an ECU. It is a combination of the severity of the Undesired Event and its frequency of occurrence, or probability.

An ECU based system need not be any safer, or any more reliable, than an existing conventional system which it replaces; however, in terms of safety it should certainly be no worse. There may, nevertheless, be higher expectations of an ECU based system than of a conventional one. An ECU based system may also be designed to provide additional functions or features which would not be available with a conventional system. These additional features may give rise to greater safety implications than those of the conventional system, and hence require higher integrity. In the past advances in the technology of automotive systems have brought subsequent regulation associated with its application.

In principle, both the risk associated with an Undesired Event and the integrity of a system are continuous quantities, i.e. in theory it should be possible to represent measurements of both risk and integrity using a real number scale. In practice, however, discrete approaches to the representation of both risk and integrity, using a number of levels for each of them, is

more manageable. This discrete approach is well developed for the categorisation of the risks in safety-related applications such as avionics and defence [4], and is proposed in the draft IEC standard [3].

There may be a number of levels of risk associated with an ECU, and these may give rise to a number of different Integrity Levels. In principle each ECU should have a single Integrity Level, but in practice different subsystems or modules of the ECU may have different Integrity Levels related to the different levels of risk with which they are associated. However, if the ECUs are on a network and can influence each other then the overall system integrity should be that of the highest constituent part (see [28]).

4.1.1 Undesired events

The Undesired Events associated with an ECU are dependent on the functionality of the ECU and the way in which it is designed to carry out its functions. It is possible to determine the set of Undesired Events which are likely to be associated with the particular type of ECU chosen.

4.1.2 Severity

The severity of the outcome of an Undesired Event will generally vary depending upon the combination of factors at the time at which it occurs. For example, an engine cut out at high speed in heavy traffic will have a greater severity than the same Undesired Event at low speed in an empty car park.
A number of different viewpoints may be taken with respect to severity. In principle, four viewpoints are possible (no order is implied):

i) harm or damage to humans, vehicle, property or environment;
ii) legal implications;
iii) disruption to normal traffic flow; and
iv) financial loss.

In most cases financial loss is associated either with harm, damage or disruption of traffic flow or failure to comply with the law. Because of this, only viewpoints (i), (ii) and (iii) need to be considered for the purposes of determining the integrity required of a vehicle system, with care being taken not to overlook viewpoint (iv).

4.1.3 Probability

The probability of a failure occurring, and then producing an Undesirable Event with any particular degree of severity, is dependent not only on the probability of the failure itself occurring, but also on the probability of the combination of other factors being such that the outcome of that failure has that degree of severity.

The Integrity Level of an ECU should be determined by assuming it has failed and then calculating the effects of all the possible Undesired Events. Thus the integrity to which the

system is produced is derived from analysing likely consequences of an assumed failure in conjunction with the probability of that Undesired Event occurring.

The way in which the probability of the occurrence of an Undesired Event is considered is dependent upon the causal relationship between the Undesired Event and the system with which it is associated. There are four possible relationships:

i) the ECU failure can initiate the event (failure to operate):
A system failure can cause an Undesired Event. For example a failure of an engine management system could cause the engine to cut out whilst the vehicle is travelling at high speed.

ii) the ECU failure can enable the event (failure on demand):
A system failure can result in the failure to take appropriate remedial action with respect to an Undesired Event initiated for other reasons. For example a failure of an anti-theft device could be such that it fails to sound an alarm after the vehicle has been broken into.

iii) the ECU failure has no effect on the event:
A system failure can be such as to have no effect on the event being considered. For example a failure of an in car entertainment system could neither directly initiate nor directly enable loss of steering. Such failures need not be considered in any discussion of that particular event provided that their independence can be demonstrated.

iv) the ECU is affected by the event:
This is not in itself a matter of concern unless the effect is such as to involve a later causal relationships of types (i) or (ii). For example, an electrical fault is not of concern (in this context) unless it leads to an ECU failure. Accordingly this type of relationship need not be considered.

In any vehicle system containing an ECU, only Undesired Events whose causal relationship with the system is of type (i) or type (ii) need be considered. Types (iii) and (iv) need not be investigated as they do not cause the Undesired Event under consideration.

If types (iii) and (iv) are subsequently connected with another event then this should be apparent from the analysis.

Relationship (i) can arise at any time but is only of concern if circumstances exist which allow a system failure to result in the Undesired Event. For example, a failure of a cruise control system could only cause a collision (the Undesired Event) if the circumstances were such that the cruise control was in use in sufficiently high traffic density. It is possible that this could always be the case, but more typically these circumstances only exist for part of the time. Hence, ideally, the probability required is the probability of the failure of the system failure combined with the probability that the circumstances exist in which system failure could result in the Undesired Event.

Relationship (ii) only arises if the Undesired Event has been initiated. Relationship (ii) events typically involve a protection system which is only required to take action if an Undesired Event occurs. Hence the probability required is the probability that the Undesired Event is initiated combined with the probability of the failure of the system to react on demand. For example, if an anti-theft system failed in such a way that it would not set off the alarm when someone attempted to break into the vehicle, allowing the Undesired Event of the vehicle being broken into to occur, the probability required would be the probability that a break-in was attempted and the system failed on demand.

4.2 Further integrity considerations

The integrity to which a system is to be produced is affected by a variety of factors. A manufacturer may wish to increase the integrity of a system in order to safeguard its reputation or to provide a marketing advantage within the industry.

As in any competitive industry it can be expected that some organisations will wish to exceed the requirements for any particular Integrity Level. Indeed, although it is currently considered that mathematical based programming methods are not yet sufficiently disseminated within the automotive industry, it is known that some automotive companies are actively increasing their expertise in this area.

Other industries which deal with safety related systems require rigorous operator training to be given. Since it would be unrealistic for the motor industry to train every driver in the correct use of its systems, the product may be produced to a higher Integrity Level in order to lower the probability of failure when the system is being used by an inexperienced operator.

4.2.1 Multiplicity of systems

The automotive industry is a volume manufacturing business and large numbers of any one vehicle are produced. This creates a different situation than, for example, a single industrial process installation.

The presence of a multiplicity of similar ECUs will increase the frequency of a failure. An increasing vehicle population increases the likelihood that when an ECU failure occurs a vehicle is experiencing the conditions which will then lead to an accident. The required integrity of the system is thus increased due to the number of vehicles produced.

If a purely quantitative approach to risk assessment was to be used, the probabilities calculated for a single vehicle would need to be multiplied by the vehicle population to establish a global figure. An annual figure is likely to be appropriate, since it could then be compared with other road accident statistics, assuming such figures are available. Such annualised figures would need to consider active testing, diagnostics and routine servicing of the considered equipment.

The vehicle population to be used would most probably be that for the particular manufacturer concerned and not for the total number of similar systems in cars throughout the entire UK vehicle population. However if identical systems were being used by more than one manufacturer then a more complex counting system might be required.

A further impact of a multiple vehicle population arises from the nature of software induced failures. Software faults are not random, but are due to design errors (systematic faults). If the appropriate conditions occur so that a software fault leads to an ECU failure then those conditions will cause a failure on each and every unit. Thus, there is a probability that a software fault may lead to several incidents before it is recognised and corrected. There is also a possibility that a latent fault may never be experienced, or that a fault which does occur may never be correctly identified.

The conditions which stimulate the fault may or may not be random, but their appearance over the life of a vehicle will be distributed and will depend on the vehicles operational profile. Any fault whose initiating conditions occur frequently will usually have emerged during testing before production, so that the remaining obscure faults may appear randomly, even though they are entirely systematic in origin.

The effect of an increasing vehicle population is to increase the severity classification associated with a particular fault due to the increasing possibility of an Undesired Event. This would place a higher obligation on the system, but since it is a common factor for all Integrity Levels, it would not affect the relative integrity of one ECU type compared to another. Comparisons with other sectors may also need to be taken into account, e.g. a series of Public Service Vehicle (PSV) accidents due to the same underlying fault are likely to be perceived to be as unacceptable as a single serious railway accident.

A quantitative approach thus needs to consider the size of the vehicle population and the likely multiplicity of incidents. One possible way of doing this is to increase the severity level in order to deal with the increased number of possible injuries or deaths. The process of allocation is inevitably subjective to some extent and relies on the judgement of the engineer concerned. Past accident data may be used if such information is reliably available, which it will not be for novel systems.

If a qualitative approach is used then the vehicle population is not relevant. In this case, the Heavy Goods Vehicle (HGV) or PSV sectors may face high integrity requirements even though the total numbers are lower than for private cars. This can be justified by the observation that their utilisation, and thus exposure, is higher, and, in the case of PSV the obligation to fare paying passengers is traditionally seen as to provide a safer mode of transport, as is reflected in other regulatory environments such as civil aerospace.

4.2.2 Decomposition into subsystems

It is difficult to design small micro-controller based systems so as to achieve total independence between separate subsystems or modules. Faults in one subsystem may therefore result in incorrect operation in another. Thus, in principle, all subsystems or modules of an

ECU residing on a single processor should have the same required Integrity Level. However, if a subsystem or module has been identified as having a lower Integrity Level than the ECU as a whole then, provided that it is shown that failure of this subsystem or module cannot compromise the integrity of the ECU as a whole, this lower Integrity Level may be used in design and assessment. It is essential that where such lower Integrity Levels are used, the justification is fully documented.

For example, if the required high integrity is provided by a relay interlock which protects against software errors, then the software subsystem may be permitted to have a low integrity (a form of safety bag). If two diverse channels provide truly independent detection and reaction to errors, then each channel may be permitted to have a lower Integrity Level than that required for the combination making up the system.

4.2.3 Risk compensation

Enhanced performance of a system due to ECU control allows a driver to rely on the system functionality and possibly to drive more dangerously than before, e.g. the reliance on Anti-lock Braking System (ABS) in icy road conditions. Note also that where an ABS system reverts to normal braking, there is still the possibility of an accident even under normal conditions. The driver may be so accustomed to ABS that he or she may still try to take advantage of the enhanced functionality even when the warning lamp is lit. Such details should be considered when assessing Integrity Levels.

5. Determining integrity

5.1 Standards

In current work on standards and safety related software a number of different approaches to the determination of risk have emerged.

IEC SC65A (Secretariat) 123 [3] defines several methods of classification, some of which are discussed in this section.

In DEFSTAN 00-56 [4], Accident Sequences are considered which consist of all the contributory effects to the accident under consideration. A set of severity levels and a set of probability levels are presented. A severity level and a probability level are then allocated to the Accident Sequence, and the risk level associated with each Accident Sequence is then determined using a table.

In DIN V19250 [5], Undesired Events associated with a system are considered. Four parameters are used, each with number of categories:

i) extent of damage;
ii) period of exposure to danger;

iii) possibility of avoiding danger;
iv) probability of the Undesired Event occurring.

The Undesired Event under consideration is allocated a category for each parameter and a requirements class is then determined for the Undesired Event using a tree structure.

In the approach used by DEFSTAN 00-56, severity and probability are the only two parameters considered, whilst in the DIN V19250 approach, two further parameters are added, namely period of exposure to danger and possibility of avoiding danger. However, these two additional parameters are accounted for in the DEFSTAN 00-56 approach, since the probability used incorporates the sequence of events which lead to the Undesired Event. Whereas the DIN V19250 approach only considers the probability of the Undesired Event, regardless of other circumstances.

5.2 Practical approaches

This section describes the three approaches that have been considered for the MISRA guidelines.

5.2.1 Standards based approach

In Appendix A a method is presented for determining the Integrity Level associated with an ECU, which is based on current generic computer system standards work, namely IEC SC65A (Secretariat) 123 [3], Interim Defence Standard 00-56 [4] and DIN V19250 [5]. These standards address risk primarily as a combination of the severity and the probability of occurrence of the events (or combinations of events) which are considered to present a hazard.

Whilst the approach taken in [3] for quantitative determination is difficult to apply to the automotive industry due to the lack of numerical data, the method in Appendix A is a possible systematic approach for deriving the probability and severity values with which to determine the Integrity Level.

5.2.2 Controllability approach

The DRIVE Project V1051 [6] took a different view of integrity which is readily fitted to the automotive industry, but which departs from the quantitative risk methodology favoured by the standards. The DRIVE Safely approach recognises that the primary reason for an automotive accident is some form of loss of control, and thus classifies ECU failures in these terms. The approach is discussed in Appendix B.

5.2.3 Pragmatic approach

Integrity levels can be approached qualitatively by associating each level with a given severity, as follows:

- *Integrity Level 4* — represent the integrity required to avoid disastrous accidents, which are unlikely to be a feature of road travel unless the concept of vehicle road trains is extended to include PSVs. In this case the high requirements appropriate to current thinking in the Nuclear or Rail Transport industries would be appropriate.
- *Integrity Level 3* — represents the integrity required to avoid serious incidents involving a number of fatalities and/or serious injuries, probably only associated with PSVs or vehicle road trains. As such a very high level of rigour is required and a considerable depth of fault tolerance would be expected.
- *Integrity Level 2* — represents the integrity required to avoid more serious, but limited, incidents some of which may result in serious injury or death to one or more persons. This Integrity Level can be achieved by rigorous methods and a fault tolerant design.
- *Integrity Level 1* — represents the integrity required to avoid relatively minor incidents and is likely to be satisfied by a certain degree of fault tolerant design using guidelines which follow good practice.
- *Integrity Level 0* — represents the integrity associated with no risk to persons and in effect represents the "don’t care" condition. However, the automotive industry is an international, high volume manufacturer and supplier and for economic reasons needs to adhere to basic good practices, whether safety related or not.

5.3 Recommended approach

It should be noted that the application of integrity is an unsolved problem in many industries. The standards bodies are still grappling with how to provide definitive guidance on how to approach integrity in a manner that is suitable for all industry sectors. The recommendations made in this section should therefore be taken with this understanding.

Three approaches for the determination of the Integrity Level of automotive systems have been presented in Section 5.2 as follows:

i) pragmatic;
ii) controllability;
iii) standards.

The above are arranged in order of increasing effort and decreasing dependence on arbitrary classification.

The pragmatic approach requires less effort than the other approaches but relies on a rigidly defined classification scheme which may be difficult to apply to novel applications.

The approach that is based on the standards provides a good focused argument and is numerically based. However the argument is not as objective as it might at first seem since

there is currently no available data to justify a particular choice of figures. At best this will lead to a lack of consensus, but at worst it might lead to a false sense of security in the feeling that one has performed an exact analysis.

The controllability approach was specifically designed to avoid the use of any figures, recognising that their values were always likely to be contentious. Instead the approach recognises that, in an **automotive** system failure scenario, between a failure and its resulting accident there is some loss of control. It is therefore possible, on occasions, for a person (usually the driver of the afflicted vehicle) to perform some action, or actions, to attempt to avoid, or mitigate the effects of, the final accident. Whilst the approach does not claim to be completely objective, experience has shown that it can produce results that can be accepted.

The recommended approach for identifying an Integrity Level is therefore to allocate a controllability category to the hazard(s) associated with a failure of the ECU. Whilst the technique does not follow the letter of the standards approach, it does follow the spirit. The following sections describe the processes that should then be performed both to achieve that Integrity Level, and to demonstrate that it has been achieved.

6. Assessment

Assessment can be first, second or third party. First party assessment is when the software is assessed by the originating organisation, i.e. the producer. Second party assessment is when the assessment is done by the organisation that is purchasing the software, i.e. the customer. Third party assessment is when the assessment of the software is done by an external independent organisation.

There are also different degrees of assessment. The simplest is a check that the development process has been followed, e.g. through confirmation that certain design material exists; this is sometime called a process audit. The most rigorous is a full assessment of the product, e.g. by checking the contents of each and every piece of design material. The word "assessment" in this document will cover all these interpretations, the degree of rigour being used at any time will increase with Integrity Level.

The degree of independence necessary will depend on the circumstances of the organisation and the Integrity Level of the system. The highest levels of integrity should require the highest levels of independence in order to ensure that there is no bias from the development team, nor misplaced pressure from management. Large organisations may be able to provide this independence internally, but sometimes it may be necessary to have an external assessor to carry out this function, especially if there is little experience in the assessment of safety-related software within the organisation (see also [31]). The third party relationship between development and assessment is indicated in Figure 1.

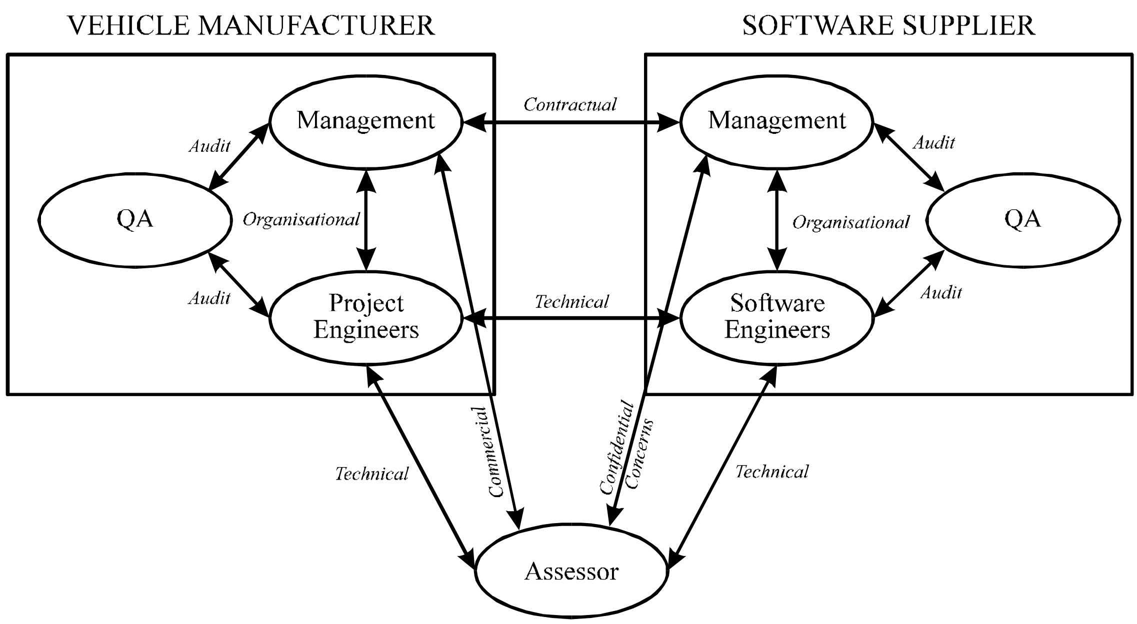

Figure 1 - Management Interrelationship

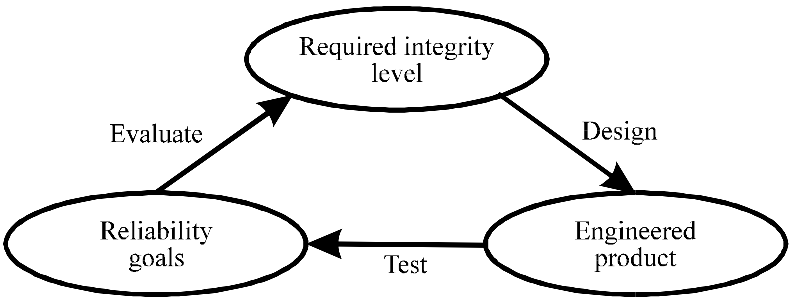

Figure 2 - Traditional Engineering

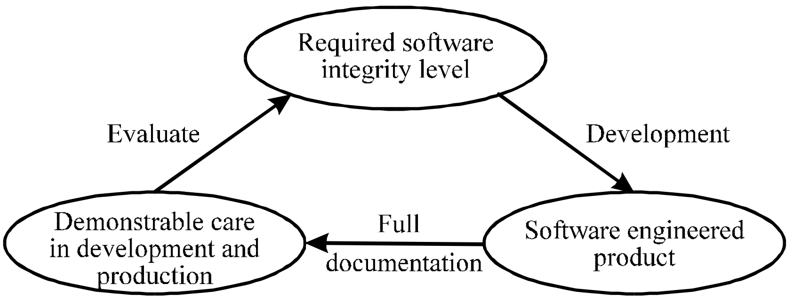

Figure 3 - Software Engineering

6.1 Confidence levels

It must be remembered that different persons will hold different perceptions of their idea of confidence in a software product, e.g.

a)	The programmer	Always runs Never crashes
b)	The designer	Always does what it is supposed to do Never does what it is not supposed to do
c)	The user	Always does what it is expected to do Never does what it is not expected to do

These perceptions are again driven by the different severity factors which arise from different points of view because software is always expected to work and never to fail. The accurate translation of the user's perception through the development process is therefore a vital factor in building confidence to a particular Integrity Level. One way of doing this is with Total Quality Management (see [7]); modelling, animation or rapid prototyping (see [31]) may also be used.

6.2 Satisfying integrity levels

Electro-mechanical systems can be engineered, by the selection of a suitable architecture and the selection of suitable components, to meet the requirements of a given Integrity Level. Failure rate is the important metric when assessing a piece of hardware which appears to be functionally correct. Using characterised standard components (with known failure rates (see [8])) and by performing Failure Mode Effect and Criticality Analysis (FMECA) [9, 10] it is possible to demonstrate that the system has a sufficiently low failure rate so that it can be certified for use at the specified Integrity Level. We have a closed loop between the required Integrity Level, the engineered product and the reliability goals as shown in Figure 2.

This model, however, assumes that all the failures are as a result of random faults due to component wear, or manufacturing defects. There is an implied assumption that once the prototypes are accepted then the design is correct i.e. there are no systematic design faults, indeed this is the basis of Type Approval.

6.2.1 Software integrity levels

The concept of a random fault, however, does not apply to software. There are times when it is possible for software to **seem** to exhibit faults in a random way, but this is always due to the hardware interacting with the software. Any given set of inputs presented to a program will always produce the same output, though it may be difficult to reproduce the input set. For example a problem can occur when, due to wear and tear, hardware begins to operate at the limits of its specification; if the software specification is incomplete then failures can begin to occur in a random manner even though they are actually due to systematic faults in the software.

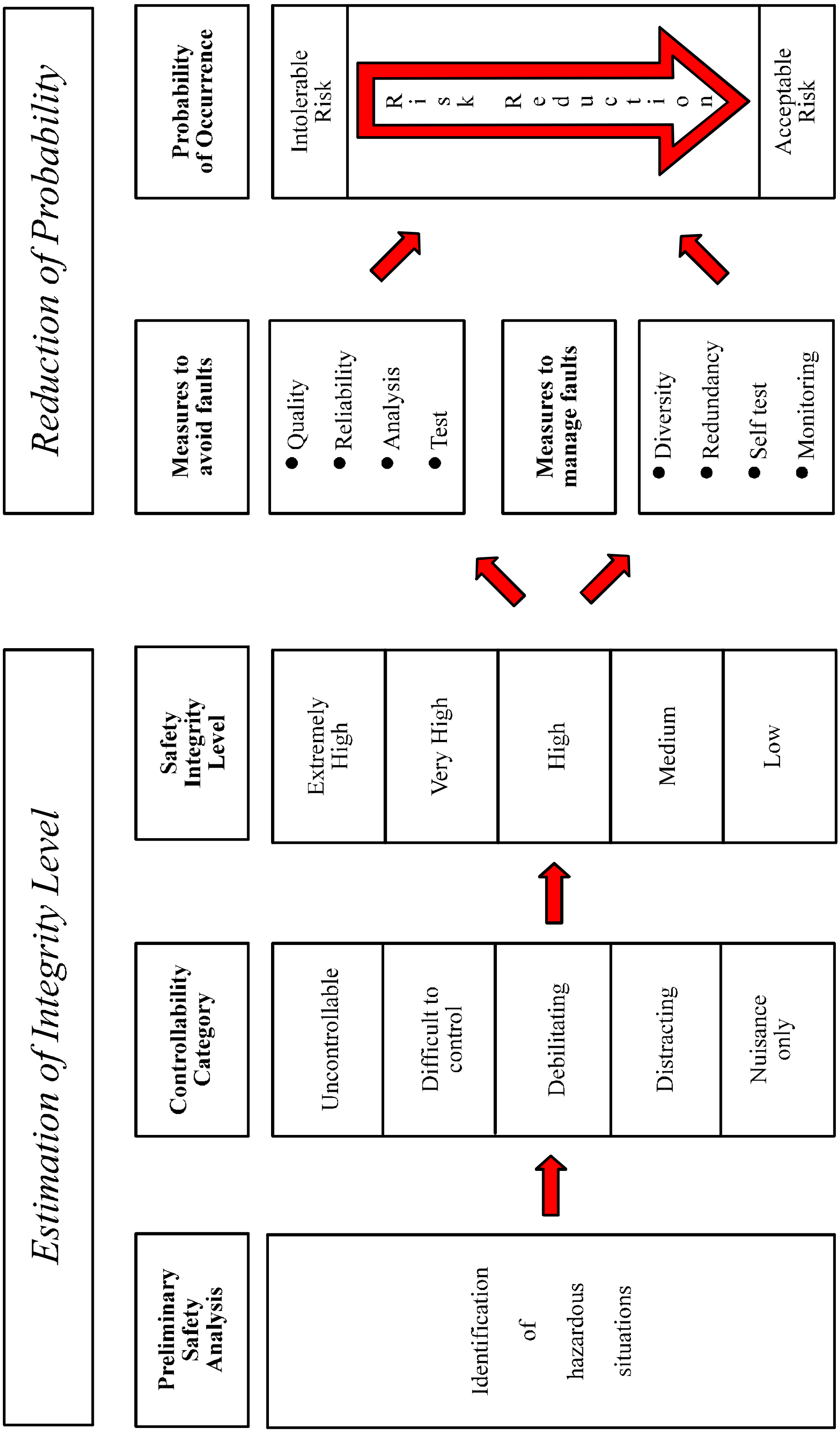

Figure 4 - Approach to avoid hazardous malfunctions

Figure 2 cannot therefore be applied to software products, and a modified approach has to be found for software assessment. Systematic faults can, of course, also occur in electro-mechanical systems, but the difference in complexity between the two types of product means that such faults are far more likely to be found during the testing of hardware, than of software.

It can be argued that with the current knowledge of software engineering a binary system is appropriate. If the function is not safety-related, then it can be produced to meet the manufacturer's required performance and/or appropriate British Standards Institute (BSI) standards. However for safety-related software the full rigour of safety-critical production, evaluation and certification should ensue to ensure that there are no latent systematic faults. Such binary classification has the great advantage of being simple but the disadvantage of being simplistic. It fails to recognise that in the real world there are few clear cut binary situations, and the situation is usually a hierarchy of "grey" scales. How then can one assess software to conform to one of five levels of confidence?

Given two pieces of software with the same functionality, how can they be tested to identify which one is the "better"? It is not proposed to give any formal definition of the word better except to note that in this context the tests should indicate that one piece of software is a more dependable product than the other for a particular task in a particular environment.

If the tests consist of solely exercising the software then, providing both pieces of software produce the same expected results in every test carried out, there is no way of prioritising them. This so-called "black box" testing is only sufficient for non-safety-related systems. Successful black box testing simply indicates that the software has produced the expected results for a specific set of test inputs. However software does not have the property of continuity possessed of hardware, such that only samples of a range of values need to be tested. Since it is extremely unlikely that all possible combinations of test inputs will be exercised, there is no way of knowing from these tests how the software will react to inputs which were not in the test set, or to inputs given in a different order. Thus testing on its own is not adequate for assessing safety-related software, and it is therefore necessary to do it by other means.

Assessment of software should be based on the totality of information made available about the software. The degree of confidence that one can have in a given piece of software in a safety-related application is a function of the quality, and the extent of available information about the specification, design, implementation and testing of the software. Assessment is concerned with gaining a level of confidence in the software that corresponds to the Integrity Level. The highest confidence levels are required of the most safety-critical products. It therefore follows that the evaluation of these products requires the fullest information about their development. In addition it is generally accepted that certain specification, design, implementation and testing techniques are appropriate for high integrity software. Assessment, therefore also has to take account of the actual techniques used at the various stages in the product's life-cycle.

We will therefore follow the analogue of Figure 2 which is shown in Figure 3. Once the

required software Integrity Level has been determined in accordance with Appendix B the software engineered product is developed. By analysing the documentation produced during the development it is possible to demonstrate care in development and production. An evaluation of this care in development and production leads to a level of confidence that should correspond to the original required software Integrity Level. This is already the basis for the Information Technology Security Evaluation Criteria (ITSEC) [11].

7. System assessment

The assessment of systems is based on the results of reviews and analyses, testing and an evaluation of how well the development was done. In this respect the procedure is similar to that shown in Figure 3; however system evaluation has to cope with both systematic faults and random faults. In this section the general approach for the assessment of systems is described, together with an overview of the measures used to avoid and to manage hardware faults.

In order to ensure the safe operation of safety-critical electronic systems, it is necessary to recognise the various possible causes of failures and to ensure that adequate precautions are taken against each one (Appendix C describes some topics which need to be considered during the development of a system). The main safety objective is to avoid the hazardous malfunction of the electronic system with sufficient probability for a particular application. This goal can be attained if the following approach is applied. During a preliminary safety analysis the hazards must be identified and evaluated. The results place each hazard into one of five controllability categories. This will then dictate the Integrity Level required of the design for the new system. According to the Integrity Level a combination of measures with suitable effectiveness must be taken to avoid and to control failures. This approach is shown in Figure 4.

7.1 Hazard analysis and traceability

In order to assign an initial Integrity Level to the design and development of a system it is necessary to perform a preliminary safety analysis on the proposal; this will also identify the high level safety requirements. As the design proceeds it is also necessary to trace each safety requirement to that part of the physical architecture by which it is implemented, as part of the assessment process. The EC DRIVE II project PASSPORT (V2057) has developed a systematic framework for performing these tasks which is summarised in Appendix D.

7.2 Risk assessment

A failure analysis should be performed on all systems to determine both the effect of the failure of any sub-system or component (e.g. using Failure Mode and Effects Analysis (FMEA) [9, 10]), and also to determine the combination of faults that could produce a failure

(e.g. using Fault Tree Analysis FTA [12, 13]). This approach is, however, not applicable to programmable and complex non-programmable systems because it is impossible to evaluate the very high number of possible failure modes and their resulting effects. One approach is to consider faults at the module level for complex electronic systems, and at the component level for interfaces, actuators, sensors, electrical and simple electronic circuits etc. (see [6]).

The objectives of the risk assessment are typically:

1. To show that no single point of failure within the system can lead to a potentially unsafe state, in particular for the higher Integrity Levels;
2. To show that the risk of multiple (sequential and simultaneous) failures that would cause an unsafe state is within an acceptable level.

Note that some vehicle faults may be permitted to remain until repaired either by the owner, or at the next regular service.

The acceptable level of risk, which should be agreed by the company involved as a matter of policy, should include considerations such as current practice, technical feasibility, customer expectations, product liability, legislation and commercial risks. The As Low as Reasonably Possible (ALARP) principle should apply [3]. The probability of the failure occurring should be taken into account. It will normally be possible to specify a cut-off level (e.g. two orders of magnitude less than the figures required for the overall system). Failures with a lower probability than this can be excluded from the analysis. Some exclusion will be necessary, as otherwise the analysis will become intractable (e.g. the possibility that the wires inside a multi-wire cable touch each other can be excluded if they are designed to a quality specified by suitable standards and they are protected against external damage).

7.3 General assessment

The number of possible failure modes covered is a characteristic of each particular Integrity Level. The coverage of failure modes can be increased by increasing the number of counter measures against failure, or by designing for fault tolerance. The chosen combination of measures must be selected in such a manner that the combination effectively reduces the probability of occurrence of an Undesired Event from an intolerable to a tolerable level. In other words there will be an overall reduction in risk in spite of the increase in complexity of the final system.

It is important to note that for Integrity Levels 0 and 1, measures against failure modes are only required for those faults with a relatively high probability of occurrence. However the higher Integrity Level systems will also require measures which take account of failure modes of faults with a low probability of occurrence.

For a particular system a combination of measures must be chosen taking into account the possible failures modes. The required rigour of the combination of measures depends on the estimated Integrity Level. Three classes of rigour are proposed: low, average and great.

The evaluation of the rigour is based on the particular combination of measures which is chosen. Specific measures are necessary to **avoid** systematic hardware faults and other measures are necessary to **control** systematic and random hardware failures. A proposal for the required rigour of a combination of measures for each safety Integrity Level is given in Table 1.

Measures to avoid and control hardware faults are discussed in Appendix E. The characteristics of the system strongly influence the selection of measures. In particular, the following should be considered:

- has the system a safe state?
- does the system return to a safe state on failure?
- is the system able to tolerate an incorrect signal?
- is it possible to detect failures that depend on information flow or system conditions?
- could faults only be detected through additional self-tests?
- what failure tolerance time is allowed?
- is it possible to examine the rigour of the measures?

Measures to		**Integrity Level**				
Avoid (Development)	**Control (Operation)**	**0**	**1**	**2**	**3**	**4**
systematic faults		low	low to average	average	average	great
	systematic failures	-	-	low	average	great
	random failures	low	low to average	average	great	great

Table 1 — Required rigour of a combination of measures against hardware failures

7.4 Analysis and test

The processes of analysis and testing should be executed in parallel with the development process to determine whether the specified safety requirements are fulfilled. A distinction must be made between the system functional safety requirements (the functions that must be carried out by the safety-related control system should a hazardous failure occur in the technical process, or in the safety-related system itself), and the system safety integrity requirements (the measures for avoiding and for controlling hardware failures). Analysis and test plans should be produced concurrently with the hardware development, and they will set out the procedures to be followed for these activities. The plans, the results, and all reports

should be documented in such a manner that they can be used for further assessment purposes.

An analysis and/or test plan should be created for each phase of the life-cycle of the system's development, and also for the validation of the final system. A report should be written after each of these various activities.

7.4.1 Analysis and test plans

The analysis and test plans should contain details of:

- the objectives of the analysis or test
- the specification of the analysis or test strategies
- inspection, mathematical and systematic analysis, module and sub-system testing, functional testing, failure mode testing, environmental and EMC testing
- the test and simulation equipment, techniques and tools
- the execution of the analysis and test procedures
- the procedures to be followed if failures are detected.

7.4.2 Testing modes

The analysis and test plan for the validation process of the final system should consider all phases of the system operation, examples of which are:

- not operational (e.g. parked, ignition off)
- standby (e.g. parked, ignition on)
- start up (e.g. engine crank)
- operational (e.g. a demand for a function)
- steady state (e.g. normal driving)
- shut down (e.g. engine switch off)
- abnormal conditions (e.g. system failure, driver misbehaviour)
- maintenance (e.g. servicing)
- periodic inspection (e.g. diagnostics)

7.4.3 Analysis and test reports

All analysis and test activities and results should be documented. The reports should collectively contain:

- the unique identification number of the specimen/document under inspection (Vehicle No., Part-No., Version-No., Serial-No., Drawing-No. etc.)
- details of the analysis and test mode
- the regulations for the analysis and the test
- the objective (aim) of the analysis and/or the test
- a description of the analysis and/or test methods, techniques and procedures
- the test conditions

- details of the analysis, test and simulation equipment/tools
- the analysis and/or test results
- details of all failures detected during the analysis and the test
- the name(s) of the inspector(s)
- the name of the person who is responsible for the analysis and test process (e.g. project manager)

8. Quality assurance plan

The benefits of quality management lie in an improvement in quality which is reflected in a reduction in costs associated with the failure of the product. This is particularly true for software, for which overruns against time and budget, as well as failure to perform to specification are quite common. The software development life-cycle should be performed under the control of a software quality assurance plan such as TickIT (ISO 9000-3). Project management and quality control cannot be applied retrospectively. If some activity in these areas has been overlooked, and its omission is a good cause for a lack of confidence in the integrity of the development process, there is normally nothing that can be done to rectify the over-sight except to do the work again. On the other hand it should be acceptable for controlled changes to be made to a piece of software in response to recommendations by the assessor.

In order to avoid any misunderstandings, the software quality plan must be documented. The plan should contain the following main items:

1. project organisation.
2. a set of planned procedures to be followed at different stages during the project.
3. a set of planned management processes to ensure compliance with the procedures in (2).
4. a set of planned management reviews of the effectiveness of the procedures in (2) and processes in (3).
5. a set of planned documentation together with their authors and readers.

The aim is to ensure that the software meets its requirements by preventing non-conformity at all stages in the life-cycle. Checklists are particularly useful in ensuring that nothing has been left out at the planning stage [14, 15, 16]. One possible plan is given in Appendix F.

9. Software assessment

9.1 Assessment of design material

There are two aspects to be considered, the nature of the design material required at a particular Integrity Level, and its contents at a particular Integrity Level. The basic nature of the design material does not vary to any significant extent with the level, however the detail

will increase (e.g. to satisfy a degree of review, enable consistency checks, etc.) with Integrity Level to match the need for more information at the higher levels (see Section 6).

The design material falls into three main groups in addition to the software code itself; plans for general activities, specifications for specific activities, and results of specific activities. The specifications and result relate to the various phases in the life-cycle of the product; the end of each phase normally being marked by a report on that phase and a specification for a future phase. A possible list of information required during the software development life-cycle is as follows, the list is not exhaustive but indicates the types of activity that should be performed, nor is it necessary for each title to be represented as a separate free-standing report (see [31]):

- Quality Management Plan
- Quality Assurance Plan
- Software Design Plan
- Software Validation Plan
- Software Design Review Plan
- Software Acceptance Plan
- Maintenance Procedures

- Software Verification Specification
- Software Requirements Specification
- Software Detailed Design Specification
- Software Module Specification
- Software Acceptance Test Specification
- Software Module Test Specification
- Software Integration Test Specification

- Software Requirements Verification Results
- Software Design Verification Results
- Software Module Design Verification Results
- Software Design Review Results

- Source Code and supporting documentation

- Code Review Results
- Software Module Test Results
- Software Module Error Incidence Results
- Software Integration Test Results
- Software Integration Error Incidence Results
- Software Validation Results

- Software Acceptance Test Results

- Owner Handbooks
- Service Manuals

9.2 Assessment requirements

In general the higher the Integrity Level the more information is required and the more rigorous are the necessary software engineering techniques. The methodology for performing the software assessment and the assurance assignment has the following general characteristics.

1. Each Integrity Level needs a degree of documentary evidence in order to have enough information to assess the quality of the production process. The contents of the relevant documents will be augmented to reflect the greater rigour needed for the higher levels, together with comprehensive traceability and consistency checking.
2. Each Integrity Level has mandatory techniques, expected techniques, and optional recommended techniques. Within the category "expected techniques" it is necessary to justify to the assessor why it was inappropriate to use a particular technique, or why the actual technique used was (will be) equivalent or superior to the expected one.
3. Each life-cycle process has certain requirements at each level. This is very similar to (1), it is just a different emphasis.

Many of the processes in the life-cycle group together naturally and these groupings will be used to indicate how the assessment requirements change with Integrity Level. In Table 2 the entries under the headings "Integrity Level 1", through to "Integrity Level 4", state what is required for that particular Integrity Level. Higher level entries include all the lower level entries. Because "Integrity Level 0" is the non-safety-related level it has not been included fully in the table. There are no assessment requirements for any of the development processes for low integrity software, beyond those normally used by the company. More detailed information can be found in Appendix G.

It is frequently the case that not all of the software that runs on a computer has been written by the developer, e.g. Operating systems, monitors, library subroutines, language run-time systems etc. In the event that a safety-related system uses such software, it is of course necessary to assess this to the same level of integrity as the program specifically written by the developer.

Development Process	Integrity Level				
	0	**1**	**2**	**3**	**4**
Specification & Design	ISO 9001	Structured method.	Structured method supported by CASE tool.	Formal specification for those functions at this level.	Formal specification of complete system. Automated code generation (when available).
Languages & Compilers		Standardised structured language.	A restricted subset of a standardised structured language. Validated or tested compilers (if available).	As for 2.	Independently certified compilers with proven formal syntax and semantics (when available).
Configuration Management:- Products		All software products. Source code.	Relationships between all software products. All tools.	As for 2.	As for 2.
Configuration Management:- Processes		Unique identification. Product matches documentation. Access control. Authorised changes.	Control and audit changes. Confirmation process.	Automated change and build control. Automated confirmation process.	As for 3.
Testing		Show fitness for purpose. Test all safety requirements. Repeatable test plan.	Black box testing.	White box module testing - defined coverage. Stress testing against livelock & deadlock. Syntactic static analysis.	100% white box module testing. 100% requirements testing. 100% integration testing. Semantic static analysis.
Verification and Validation		Show tests: are suitable; have been performed; are acceptable; exercise safety features. Traceable correction.	Structured program review. Show no new faults after corrections.	Automated static analysis. Proof (argument) of safety properties. Analysis for lack of livelock and deadlock. Justify test coverage. Show tests have been suitable.	All tools to be formally validated (when available). Proof (argument) of code against specification. Proof (argument) for lack of livelock and deadlock. Show object code reflects source code.
Access for assessment		Requirements & acceptance criteria. QA & product plans. Training policy. System test results.	Design documents. Software test results. Training structure.	Techniques, processes, tools. Witness testing. Adequate training. Code.	Full access to all stages and processes.

Table 2 — Summary of Requirements (see Appendix G for details)

9.3 "Non-standard" development

Most of the contents of these guidelines assume that development will be undertaken in accordance with a standard life-cycle (see Appendix H). On occasions this may not be possible, and yet it is necessary to have an assessment process to cover these situations even though the ideal set of information is not available. The following sections describe three possible scenarios when it might be necessary to make an assessment of a system in a manner that is not in strict accordance to these guidelines, and then suggest some solutions.

9.3.1 Complex environment

Sometimes it is impossible to produce a full requirements specification e.g. when the physics of the system under development is not sufficiently understood, and thus the developer is unable to predict the required input/output relationships. The full specification is, in effect, produced during the development by performing a protracted set of experiments and tests on a series of prototypes. In this situation the amplification of the specification must be made under a configuration management scheme to ensure that it always reflects the current knowledge.

9.3.2 Independent development

On occasions a vehicle manufacturer may wish to incorporate a system produced by another manufacturer, but is unable to obtain any details of the development process necessary to assess whether it can be used at a particular Integrity Level. Sometimes it may be possible to arrange for a third party independent assessor to act as an intermediary, and thus alleviate the problem.

9.3.3 Changing environment

The objective of having a hierarchy of Integrity Levels is to enable a developer to produce a system with a cost that is commensurate with the safety hazards associated with it. However, if at a later stage the usage of a system changes, then it is possible for the required Integrity Level to rise. The developer, however, will wish to use the existing system, possibly with minor alterations, without having to go through the expense of re-developing the system from scratch.

A variation on this scenario is the desire to continue to use systems that were developed before the advent of these guidelines.

9.3.4 Assessment of "non-standard" systems

There are several possible solutions to the problem of assessing systems that either cannot, or have not, been developed in accordance with these guidelines. The basic principal, however, remains the same i.e. the assessor is looking for information to build up a level of confidence in the system corresponding to the Integrity Level identified during the hazard analysis.

1. For relatively small systems it is sometimes possible to perform a 100% test on the complete system. It should be noted, however, that whilst it may be possible to perform a 100% test of the data in memory, it will not be possible to perform a 100% test on the computer system itself (i.e. processor, data channels etc.).
2. A "safety-bag" of the required Integrity Level can be put around the system. The purpose of the safety-bag is to limit the consequences of any undesired action by the system to an acceptable level. Safety-bags can often be very simple, and thus relatively easy to assess to a high Integrity Level. They are a common method of reducing the risks associated with very complex systems e.g. ones using artificial intelligence.
3. It may be possible to retrospectively validate the system, but it should be noted that this may well cost several times as much as validating an equivalent system which has been developed to the required standards. The first step should be to agree the criterion for whether or not the system will be suitable for the desired application. The approach will require a detailed assessment and analysis of the system, including the reverse engineering of any missing documentation. The assessment should compare how well the actual software conforms to modern development standards. Additional validation or verification (testing or static analysis) is also likely to be necessary. A realistic assessment of how much confidence can be placed in the operating history (if any) should be made. Pessimism should be used when accounting for different operating environments or changes to the system. Common sense must be used in deciding which faults should be fixed (e.g. where a real fault exists) and which ones may be left in, perhaps with the provision of warnings for future maintainers (e.g. where the code is correct, but deviates from modern standards). A final safety review should be held, which considers the results of the hazards analysis, the retrospective validation, any corrections made and the reverse engineered documentation.

Appendix A — Systematic approach of determining integrity level

A.1 Introduction

The approach presented below for determining the appropriate Integrity Level for an ECU is based upon the methods described in [4] and [5] adapted to be more specific to the automotive industry. The method is divided into several stages as follows:

1. For each ECU function identify the Undesired Events;
2. For each Undesired Event, determine the Consequences;
3. Determine the Severity of each Consequence;
4. For each Consequence, determine the Probability Level using the parameters Exposure of Danger and Avoidance of Danger;
5. For each Consequence, classify the Risk Level;
6. For each ECU function, determine the Integrity Level.

We will now expand on each of these steps and conclude with an example.

A.2 Identification of undesired events

The Undesired Events stem from an analysis of the overall hazards which are being assessed. Fault Tree Analysis (FTA) [12, 13] can be used to breakdown each hazard into a series of base events which cause the hazard to occur.

FTA begins with a list of all Undesired Events. Each Event is analysed for all possible causes. These causes may also be the result of "lower" causes and so the process is repeated until base events are reached. A base event is one which cannot be divided further and which contributes, or causes, the top event to occur.

An advantage of an FTA is that it can be started before detailed design information is known. However all information must be known before it can be completed.

FTA enables a subjective hazard to be explicitly defined in a structured manner in relation to its causes. Probabilities may be added to a fault tree in order to assess the likelihood of the hazard occurring.

However, FTA relies on the experience of the engineer performing the analysis in order that all possible branches/scenarios of the tree are investigated.

A.3 Determination of hazard consequences

A.3.1 Accident sequence

An Undesired Event together with a set of external factors which may lead to an accident can be considered to be an Accident Sequence [4]. Each external factor being an event or condition which may occur as, or following the Undesired Event.

The principle of an Accident Sequence is such that in the event of a failure occurring, the circumstances and subsequent events that lead to an accident can be determined.

It is important to consider carefully the Accident Sequences applicable to any ECU and not to assume that they will be the same as for some other system, however similar. Experience with similar systems can, however, be used for guidance.

Even in present systems mechanical failures can, and do, occur. The outcome of a failure may be an accident depending on the prevailing circumstances and the capacity for manoeuvre remaining with the driver.

Some factors which may affect the outcome of an Undesired Event, and which should be considered when determining the Consequences associated with a particular Undesired Event, are:

i) road conditions;
ii) traffic conditions;
iii) weather conditions;
iv) amount of daylight or street lighting;
v) speed of travel;
vi) warning time given (of Undesired Event);
vii) residual control available;
viii) physical protection available; and
ix) physical structure and condition of vehicle.

The first four of these factors are independent of the vehicle and type of system involved. The fifth factor, speed of travel, is dependent not only on the speed limit applicable to the road, but also on the speed limit applicable to the vehicle on that road, the driver and the vehicle. The last four factors are dependent upon the vehicle and/or the system concerned.

Each mode of failure of a particular ECU must be considered in determining possible Accident Sequences. However, the probability of the ECU failure is not included. The probability and severity of the accident should the ECU fail in this mode, determines the integrity required of the ECU. Where a failure rate can be quantified this can be an indicator of whether the required hardware integrity has been achieved, but software failures cannot be quantified and other measures are necessary.

An Accident Sequence can be expressed using a logical English statement, an Event Tree [17]

or Cause Consequence Graph [18].

A logical English statement, although accurate and concise, can be difficult and confusing to interpret. Event Trees and Cause Consequence Graphs are easier to interpret and check due to their diagrammatic nature.

A.3.2 Failure mode and effects analysis

A Failure Modes and Effects Analysis (FMEA) [9, 10] can also be used for determining the consequences of the Undesired Events. It is a textual approach which lists all constituent parts of a system and for each one gives its potential failure modes. The effects of each failure mode are determined and then the significance of these effects are stated. Acceptable and unacceptable failures are then apparent.

The same factors which are considered in an Accident Sequence are also considered in an FMEA. An advantage of using an FMEA approach is that the significance of Events are clearly defined. Although not in diagrammatic form an FMEA is still relatively easy to interpret.

A disadvantage of an FMEA approach is that usually, depending on objectives, low level design details must be known before a start can be made. Low level design changes should therefore be analysed in updates to the original FMEA.

A.4 Severity of consequences

In considering the severity of a possible Undesired Event it is necessary to recognise that a vehicle accident causing serious injury is as likely to cause one or more deaths. While it would be formally correct to extend an Accident Sequence or FMEA to take account of the additional variables this is considered to be unduly complex and instead the severity category is broadened to include these cases.

Each Consequence of each Undesired Event should be allocated a Severity Level, based on Table A.1.

The Severity Levels are described here in terms of harm to people. Incidents causing environmental damage are not considered alone, but can be considered in terms of the potential long-term effects of that damage. Incidents which cause a failure to comply with a legal requirement are again not considered. Such an incident can be considered to have the level of severity which would be appropriate to the type of harm or damage which the law is intended to prevent. This method of considering legal requirements is appropriate because most legal requirements are introduced with the intention of enforcing safety measures or reducing environmental damage.

Severity Level	Description	Number of Recoverable Injuries	Number of Deaths per Incident
S0	No harm or damage, some inconvenience.	0	0
S1	Minor incident, recoverable injuries, but no serious injuries (i.e. with permanent effects). Any minor accident.	<5	0
S2	Limited accident involving a few fatalities and/or serious injuries (i.e. with permanent effects). Typical single or two vehicle accidents.		<5
S3	Serious accident involving a number of fatalities and/or serious injuries. Typical accident of PSV vehicle type.		<50
S4	Disastrous accident involving many fatalities and serious injuries. Beyond the range of credible vehicle accidents except, possibly in the future, accidents involving a road-train of PSVs.		>50

Table A.1 — Severity Levels

A.5 Probability levels

The Probability associated with a Consequence may be expressed quantitatively in terms of a table, see, for example Table A.2.

The probability under consideration here is the probability of occurrence of a given Consequence, not that of the underlying ECU failure.

Probabilities as such are not quantified, instead quantified values are provided for the expected number of occurrences per annum unless corrected by maintenance or recall. No quantification is given for expected number of occurrences during the design life, because the design life of vehicles varies according to manufacturer, model, model year, etc., and so it would be difficult to use this as a standard measure.

Probability Level	Description	Occurrences Per Annum
Frequent	Likely to occur regularly	100
Probable	Likely to occur often	10
Occasional	Likely to occur several times	1
Remote	Likely to occur some time	10^{-1}
Improbable	Unlikely to occur	10^{-2}
Implausible	Very unlikely to occur	10^{-3}

Table A.2 — Probability Levels

As there is a considerable difference in the number of miles travelled per annum from one type of vehicle to another, it may be more useful to use, for example, "occurrences per 12000 miles travelled", rather than occurrences per annum.

It is difficult to use Table A.2 to assign Probability Levels. The necessary quantitative data is not available, and a qualitative assessment will be very subjective, requiring considerable justification for each section.

The assignment of a Probability Level is made more accurate by subdividing it into further steps by considering, at the time of the Undesired Event, the extent to which the driver, passenger or pedestrians are then exposed to danger, and the opportunity each then has to avoid that danger. This allows a more objective view to be taken of the overall Probability Level associated with the Consequence.

A.5.1 Exposure to danger

The parameter Exposure to Danger takes into account the time spent within the area of danger and the frequency of exposure to the danger.

To determine the Exposure to Danger it is necessary to identify the possible prevailing road, weather etc., conditions which would be necessary for the particular ECU failure to lead to the particular accident Severity Level being considered.

Exposure to Danger is the relative probability that the vehicle and driver are exposed to those conditions if the ECU fails at any time. The probability of ECU failure is not included.

Suitable values for the choice of Exposure to Danger may be obtained by considering the proportion of time/mileage in conditions, which, if the ECU fails in the mode under consideration, would lead to the severity of accident considered. Exposure to Danger (E) may then be determined from Table A.3.

Thus, for example, considering the failure of an ECU controlling vehicle suspension, a particular failure mode may lead to breakaway and skidding. For this to lead to an accident

with Severity Level S2 then a private car may need to be exceeding 50 mph while rounding a bend of 250 metre radius, on a road where the resulting skid may cause collision with a fixed object, or with sufficient traffic to cause collision with another vehicle. These conditions probably exist in the normal use of the vehicle for 0.1% of the time or less, giving

Exposure to Danger	Description	Proportion of Time/Mileage Exposed
E1	Frequent	1 in 10
E2	Occasional	1 in 100
E3	Remote	1 in 1000

Table A.3 — Exposure to Danger

an Exposure to Danger of E3.

If the vehicle was an all purpose, 4x4 vehicle, the speed threshold would probably be lower and its normal use would take it off road where collision or equivalent danger would be higher than a private car, so that the Exposure to Danger may rise to E2.

A.5.2 Avoidance of danger

The parameter Avoidance of Danger takes into account all the factors which affect the ability of the driver, passenger or pedestrian to take some kind of evasive action in order to avoid danger, that is, to avoid or avert an accident or to reduce the severity of an accident.

To determine the Avoidance of Danger it is necessary to consider what opportunity the driver and passenger has to avoid the consequence of the failure by use of residual control, or escape, should the specific failure occur under the conditions which have been assumed in the Accident Sequence. Avoidance of Danger (A) may then be determined from Table A.4.

The combination of the parameters Exposure to Danger (E) and Avoidance of Danger (A) allocated to an Undesired Event allows the level of probability associated with that event to be determined. If the assigned probabilities P_E and P_A for parameters E and A respectively are used to determine the probability of all possible combinations of E and A we get the ranking of Table A.5 which can then be reduced to Table A.6.

Avoidance of Danger	Description	Probability of Accident Occurring
A1	Unlikely to avoid danger, inadequate residual control available	1
A2	Possible to avoid danger, basic control still available, but restricted	1 in 10
A3	Probably avoid danger, normal (un-enhanced) control available	1 in 100

Table A.4 — Avoidance of Danger

E	P_E	A	P_A	$P_E \times P_A$	Probability Level
E1	10^{-1}	A1	1	10^{-1}	P1
		A2	10^{-1}	10^{-2}	P2
		A3	10^{-2}	10^{-3}	P3
E2	10^{-2}	A1	1	10^{-2}	P2
		A2	10^{-1}	10^{-3}	P3
		A3	10^{-2}	10^{-4}	P4
E3	10^{-3}	A1	1	10^{-3}	P3
		A2	10^{-1}	10^{-4}	P4
		A3	10^{-2}	10^{-5}	P5

Table A.5 — Derivation of Probability Levels

	A1	A2	A3
E1	P1	P2	P3
E2	P2	P3	P4
E3	P3	P4	P5

Table A.6 — Determination of Probability Level

A.6 Classification of risk

The determination of Risk for each Undesired Event from the assigned Probability Level and Severity Level can now be performed using Table A.7.

	S4	S3	S2	S1	S0
P1	A	B	B	C	D
P2	A	B	C	D	E
P3	A	B	C	D	E
P4	B	C	D	E	E
P5	B	C	D	E	E

Table A.7 — Determination of Risk Level

A.7 Determination of integrity levels

Risk and Integrity cannot be considered as being the same thing. The required integrity applies to an ECU, whereas risk applies to an Undesired Event or Consequence, of which several may be associated with a single ECU. Table A.8 provides the means of determining the Integrity Level from the Risk Level(s) determined for the Undesired Events or Consequences associated with the ECU.

Integrity Level	Definition
4	One or more Risk Level A
3	One or more Risk Level B
2	One or more Risk Level C
1	One or more Risk Level D
0	One or more Risk Level E

Table A.8 — Determination of Integrity Levels

A.8 Example of integrity determination

The following is a hypothetical example to illustrate the method.

A.8.1 Anti-lock braking system on a PSV

The Undesired Events associated with ABS are:

i) complete loss of braking;
ii) uneven braking (loss or reduction of braking on some wheels);
iii) under-steer or over-steer; and
iv) loss of control.

Undesired Event (iv), loss of control, could occur as a result of loss of braking (i) or uneven

braking (ii).

For the purposes of this example, only Undesired Events (i) and (ii) are considered.

A.8.1.1 Loss of braking

R1 When the vehicle is travelling at high speed, in heavy traffic, and in the event of a need to slow down quickly, loss of braking could result in an accident which could kill or seriously injure the occupants of both the affected vehicle and any other vehicle, or pedestrian, in its path at the time of the failure. Since in this case the affected vehicle is considered to be a PSV, according to Table A.1, a Severity Level of S3 is considered to be appropriate.

R2 In order for the ECU failure to result in accident of Severity S3, it is necessary for certain conditions to be true, for example high speed travel with a requirement to slow down quickly, travelling downhill with a requirement to slow down or stop. In order for a collision to occur, there must be another vehicle or a stationary object in the path of the affected vehicle, which cannot be avoided by means of a change of course. Such conditions are likely to occur frequently, so according to Table A.3 an Exposure to Danger parameter of E1 is considered to be appropriate.

R3 If all braking capability is lost, there will be no residual control over the braking system. However, there will be some residual braking available by using the parking brake. If the vehicle is travelling at a high speed or down a steep hill, this is likely to be of limited use. According to Table A.4 an Avoidance of Danger parameter of A1 is considered appropriate for the Accident Sequences described above.

Using Table A.6 parameters E1 and A1 give rise to Probability Level P1.

Using Table A.7, a Probability Level P1 and Severity Level S3 give rise to Risk Level B for loss of braking.

A.8.1.2 Uneven braking

R4 Uneven braking could give rise to directional instability when braking, that is, the vehicle could deviate from the path which the driver is steering. When the vehicle is travelling in heavy traffic, uneven braking could result in an accident which could kill or seriously injure the occupants of both the affected vehicle and any other vehicle nearby at the time of the failure. Since the affected vehicle under consideration is a PSV, according to Table A.1 a Severity Level of S3 is considered to be appropriate for uneven braking.

R5 In order for the ECU failure causing uneven braking to result in an accident of this type, with this degree of severity, its Accident Sequence must include the condition that heavy traffic is present at the time when the failure is realised, that is when the driver first tries to use the braking system after the failure has occurred. Therefore

a situation where the vehicle needs to slow down is also required. Such a combination of conditions is likely to occur frequently, so according to Table A.3, an Exposure to Danger of E1 is considered to be appropriate.

R6 In the case of uneven braking, some residual braking ability will be available (in addition to the parking brake). According to Table A.4, an Avoidance of Danger of A2 is considered appropriate.

Using Table A.6 risk parameter classes E1 and A2 give rise to Probability Level P2.

Using Table A.7, a Probability Level P2 and Severity Level S3 give rise to Risk Level B for uneven braking.

A.8.1.3 System integrity level

The results of the above analysis are summarised in Table A.9.

Using Table A.8 the resulting required Integrity for an Anti-lock Braking System on a PSV is Integrity Level 3.

Argument Rationale	**Severity (Table A.1)**	**Exposure to Danger (Table A.3)**	**Avoidance of Danger (Table A.4)**	**Probability Level (Table A.6)**	**Risk Level (Table A.7)**
R1	S3			P1	B
R2		E1			
R3			A1		
R4	S3			P2	B
R5		E1			
R6			A2		

Table A.9 — Anti-lock Braking System on PSV

Appendix B — The controllability approach

B.1 Applicability

The Controllability approach was originally developed by the project DRIVE Safely as part of its proposal for the development of safe road transport informatic systems [6]. Such systems might be in-vehicle, multi-vehicle, at the road-side or a combination. Some systems will replace or enhance existing applications, but many will provide novel functions. The approach present here has been adapted and extended for vehicle systems only, although it does include multi-vehicle systems.

B.2 The basic philosophy

The philosophy of Controllability was developed from the usual definition of risk

$$\text{risk} = \text{effect} \times \text{probability } (R = E \times P)$$

where P is the probability of a failure - that is the situation that the system deviates from the specification - and not the probability of an accident resulting from that failure. This definition states that, in order to keep the risk at an acceptably low level, it is necessary to have a very small probability of a hazard with a large effect, or vice versa. It is therefore necessary to judge the effect of a failure.

For automotive systems the effect of a failure is by no means certain, and any attempt to put a numerical value to the probability of an accident occurring after a failure is at best highly contentious, and for novel systems extremely difficult to justify due to lack of previous experience. In order to ameliorate the process of assessing the effect of a failure the concept of Controllability Categories were developed.

B.3 Controllability and integrity levels

During the early phases of a development a preliminary hazard analysis must be performed on the system in order to identify all the hazards that are associated with it (see Appendix D). By identifying the controllability of the safety of the situation after a failure a Controllability Category can be assigned; this will then define the Integrity Level necessary for the development of that part of the system.

Note:

- We emphasise that it is the controllability of the safety of the situation after a failure that is being categorised, not necessarily just what the driver of the vehicle can do, nor necessarily just the controllability of the vehicle.
- The word "occupant" below includes anyone who has the potential to control

the vehicle systems.

- When performing the hazard analysis the emphasis is on identifying the most likely outcomes of a failure. It will usually be appropriate to ignore very unlikely outcomes ("strange scenarios"). However care should be exercised, because a small number of such unlikely scenarios with severe outcomes may form a significant risk against a set of very likely outcomes of minor severity. In addition a particular company may wish to protect a system against a "strange" scenario that has made the headlines on one occasion only, but which all future motoring correspondents are likely to test!
- The hazard analysis is first and foremost concerned with failures of one or more systems. Misuse of the vehicle is not normally considered, though the degree to which this advice is applied must be decided as company policy.

B.3.1 Controllability categories

The five Controllability Categories are defined as follows:

Uncontrollable

This relates to failures whose effects are not controllable by the vehicle occupant(s), and which are most likely to lead to extremely severe outcomes. The outcome cannot be influenced by a human response.

Difficult to Control

This relates to failures whose effects are not normally controllable by the vehicle occupant(s) but could, under favourable circumstances, be influenced by a mature human response. They are likely to lead to very severe outcomes.

Debilitating

This relates to failures whose effects are usually controllable by a sensible human response and, whilst there is a reduction in the safety margin, can usually be expected to lead to outcomes which are at worst severe.

Distracting

This relates to failures which produce operational limitations, but a normal human response will limit the outcome to no worse than minor.

Nuisance Only

This relates to failures where safety is not normally considered to be affected, and where customer satisfaction is the main consideration.

B.3.2 Integrity levels

These five Controllability Categories are related directly to five levels of integrity, i.e. failures that result in an uncontrollable hazard must be developed in accordance with Integrity Level 4, etc. The definitions of the five Integrity Levels are as follows:

Integrity Level 4

A system created to this level of integrity will give confidence to the developer and user that the likelihood of "uncontrollable" failures is extremely improbable.

Integrity Level 3

A system created to this level of integrity will give confidence to the developer and user that the likelihood of a "difficult to control" failure occurring is very remote.

Integrity Level 2

A system created to this level of integrity will give confidence to the developer and user that the likelihood of a "debilitating" failure occurring is remote.

Integrity Level 1

A system created to this level of integrity will give confidence to the developer and user that the likelihood of a "distracting" failure occurring is unlikely.

Integrity Level 0

A system created to this level of integrity will give confidence to the developer and user that the likelihood of a "nuisance only" failure occurring is no worse than reasonably possible.

B.3.3 Usage

A summary of the relationship between the Controllability Categories, Integrity Levels and failure rates is shown in Table B.1, but note that the names given to each of the Controllability Categories have been chosen for their convenience, they should always be associated with their definitions above when they are being used.

There are three classes of functionality associated with a vehicle:

i) control of, or assistance with, the main driving task.
ii) control of, or assistance with, occupant comfort or convenience.
iii) service activities.

Since the most hazardous situations are associated with class (i) the technique of assigning a Controllability Category to a failure has been oriented towards hazards in this class. Experience, however, has shown that the technique can also be applied to many hazards in classes (ii) and (iii). On the occasions when this is not possible it is recommended that category choice should be based on the severity of the most likely outcomes as specified in the above definitions, until further research has been done in this area. It should be noted that normal Health and Safety at work regulations may also apply to functions in class (iii).
It should be noted that once an Integrity Level has been assigned, the designer may wish to develop the system to a higher Integrity Level to take account of other factors e.g. legislation, environmental or commercial issues.

Controllability Category	Acceptable Failure Rate	Integrity Level
Uncontrollable	Extremely Improbable	4
Difficult to Control	Very remote	3
Debilitating	Remote	2
Distracting	Unlikely	1
Nuisance Only	Reasonably Possible	0

Table B.1 — Summary of relationship between Controllability Categories, Failure Rates and Integrity Levels

B.4 Example of controllability assignment

Figure B.1 attempts to illustrate the relationship between automotive hazards and controllability. The complexity of the mapping between hazards, severity factors and other considerations in the determination of controllability category is clear from the diagram. This section is an example of the application of the techniques discussed above. The example shows that determination of controllability category is feasible, although it is a largely subjective process.

The scenario to be investigated is "Total Engine Failure". The assumption here is that all other features of the vehicle are not affected significantly in the short term. In particular it is possible to disengage the gears, and the driver is able to operate the brakes and steering as required until the vehicle comes to rest.

The tables in Figure B.1 provide topics which are all relevant to the decision making process of allocating a controllability category for a particular system. These factors may be considered in isolation or, more often, will be weighed one against the other. There will be a number of ways of using them and experience will need to be obtained in order to discover the best one. Normally the allocation of a controllability category will be undertaken in a meeting of various experts including, ideally, both engineers and those with a knowledge of human factors, but the following paragraphs indicate a possible scenario.

One promising way of using the tables in Figure B.1 is to split the process of assessing the Controllability Category into two phases:

1. Identify how important to the control of the overall safety of the situation is the feature whose failure is being considered. For this part of the assessment two factors must be considered; the effect on the host vehicle, and the direct effect (if any) on the other vehicles in the surrounding area as a result of (the lack of) some form of communication.
2. Assess the degree of residual control over the safety of the situation that remains. For this purpose there are two more factors to be considered: the availability of suitable

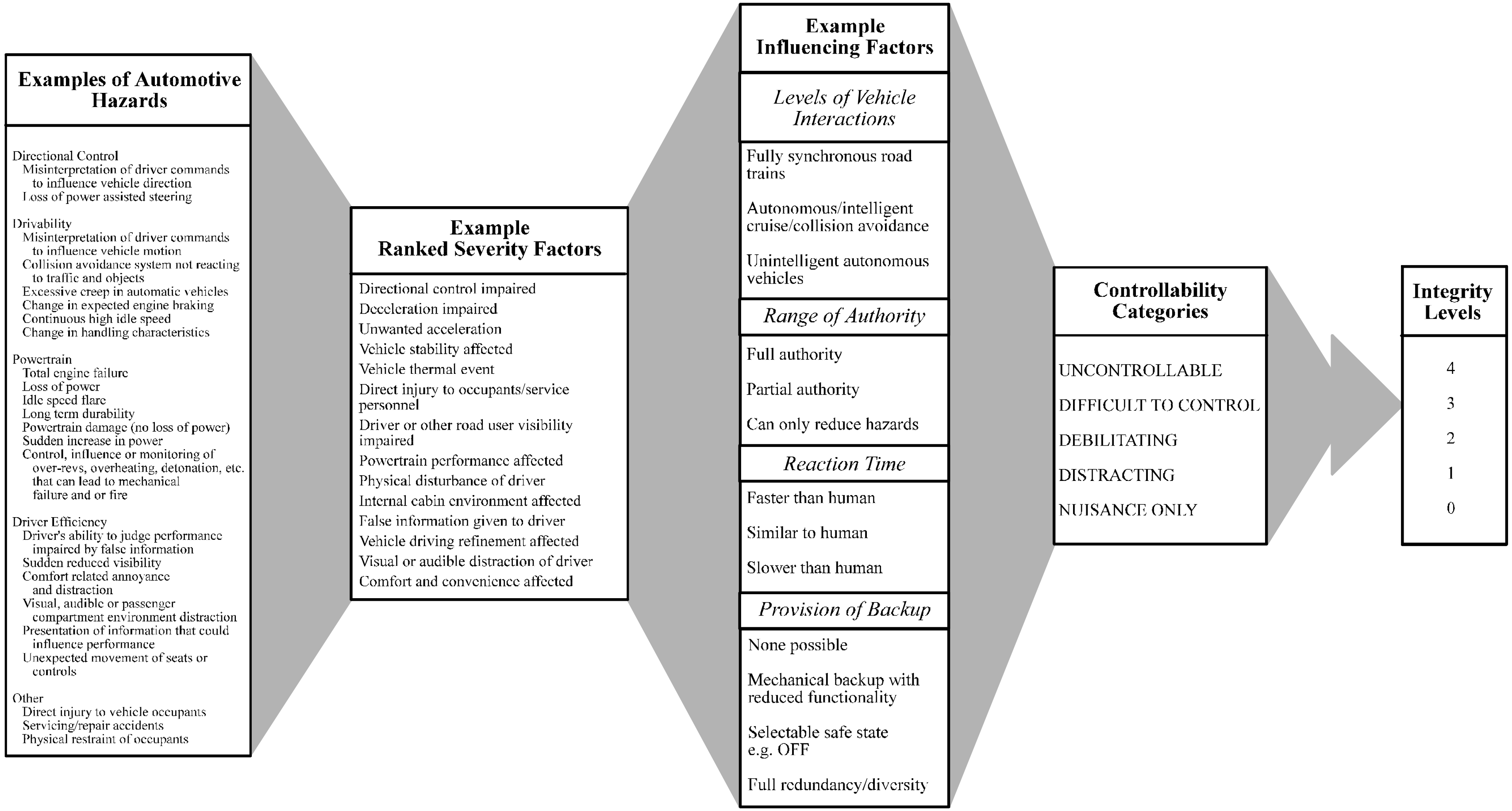

Figure B.1 - Guide to Assigning Integrity Levels

facilities, and the speed with which the driver must make use of them.

A total engine failure is now considered using this method.

- As we are assuming that the vehicle is autonomous the "levels of vehicle interaction" factor can be ignored in this case.
- An engine has considerable control over the movement of a vehicle, although this control is managed by the driver. i.e. the "range of authority" is partial. Thus since a total engine failure will remove the facility for forward acceleration, a primary function of the vehicle, we can expect a medium to high loss of control though the final value of the controllability category will depend on the next two factors.
- Although forward acceleration is lost, the driver will still be able to decelerate under control with the brakes, and have full lateral control with the steering, i.e. the "provision of backup" is with reduced functionality. This would seem to indicate that the driver should be able to avoid an accident under favourable circumstances provided this is supported by the fourth factor.
- Whilst the driver will have to react immediately to a total engine failure, this should be within his or her normal capability, i.e. the "reaction time" is similar to that of the driver. This reinforces the supposition above.

The above argument would seem to indicate that a controllability category of debilitating (failures whose effects are usually controllable by a sensible human response and, whilst there is a reduction in the safety margin, can usually be expected to lead to outcomes which are at worst severe) is correct for a total engine failure.

Appendix C — System design issues

The following sub-sections describe topics which need to be considered during the development of a system; the list is by no means exhaustive.

C.1 Sensor technology

A sensor is a very specialised piece of hardware to measure the value of one or more parameters of either the system, or of the system environment. A sensor uses a physical phenomenon to convert the quantity to be measured into another domain (usually electrical); the phenomenon, conversion principle, parameters and constraints must be properly understood and defined. The reliability of the measured value is sometimes dependent on the regular calibration of the device; the calibration technique and the procedure must be described and the calibration interval must be known. A procedure for the detection of a malfunction of the sensor is necessary, and such a malfunction should be detectable within a predetermined time period. Also, in the case of binary sensors, the likelihood of false-negative and false-positive readings should be specified. If, for reasons of diversity either more than one sensor of the same type, or more than one type of sensor is used for the measurement of the same quantity, then rules for data reconciliation are necessary. For example, under what circumstances are fundamental differences between the results of multiple sensors detected. Actuators should also be documented in a similar manner.

C.1.1 Computational procedures

The data that is derived from a sensor is not necessarily exactly what is required, as in most cases there is a difference between the sensor data and the information needed. This means that the system essentially requires the sensor data to be processed in some way. This processing can involve logical operations, some mathematical or statistical computation, or perhaps some combination, of the sensor data with fixed data (e.g. an offset) or data from other sources, including other sensors; self calibration is also possible. Application of these processes leads to some kind of conclusion, usually a value - the information that is required within the system. There is always some chance that the value presented deviates somewhat from the actuality in the real world; this must be identified at a higher level and, if appropriate, be incorporated in the safety requirements of the system (this also applies to values taken from those sensors that can provide false-positive or false-negative signals).

C.2 Feedback

The feedback of results, both raw and processed, is common in many systems. Feedback of information greatly increases the stability of a system, but only by using extra functionality. The increase in performance of a system due to feedback is thus a trade-off between the extra functionality needed and the possible increase in overall failure rate. The method of feedback chosen must match the required Integrity Level in accordance with Sections 7.2 and 7.3 (see

also [29]).

C.3 EMC

The European Commission has enacted a generic EMC directive, 89/336/EEC, covering all electrical and electronic systems sold or brought into service within the Community. Its requirements are that systems must have an inherent level of immunity to function correctly in their intended environment, and must not adversely effect the operation of other electrical or electronic systems within that environment. The only exemptions are for equipment covered by an existing product specific directive.

The Association des Constructeurs Européens d'Automobiles (ACEA) has proposed an extension to the existing product specific directive 72/245/EEC to cover both immunity and emissions of vehicles and sub-systems. This proposal (III/4127/91-EN) is currently being considered by the Commission and it is hoped it will be implemented by 1st January 1996 to coincide with the end of the transitional period on the general directive. If the proposal of the ACEA is not implemented by that date the technical construction file (TCF) route will have to be used by automotive manufacturers to demonstrate compliance with the EMC directive.

In the UK, the generic EMC directive is implemented by *The Electromagnetic Compatibility Regulations*, UK Statutory Instrument 2932 (1992).

C.4 Communication facilities

A system may communicate with other systems (data communication, transfer of materials, (electro) mechanical actions, energy transfer, etc.), with humans (operators, users) or with collections of data or material. Data communication with other systems requires the data to be consistent at the logical level within the systems involved, the communication procedures to be fully defined, the error procedures to be defined and that system availability is guaranteed (see also [28]).

Human communication with a system does have some relationship with human behaviour and human error (see Section C.8), but this is mainly in the area of ergonomics and of Human Computer Interaction (HCI).

C.5 Data-bases

Although there may not be a requirement for having one or more in-vehicle data-bases, they will exist as part of many support systems e.g. service testing, mid-life update information etc. Communication with data-bases also requires logical coherence and well defined procedures. Data retrieval from a database is sometimes straightforward, but on occasions it

is not easy to guarantee that the received data is indeed identical to the requested data. As well as "normal" faults in the associated software, data-bases can exhibit a form of ageing whereby the structure increases in complexity over time; in addition "deleted" data can sometimes still be accessible. In the event that severe time constraints have to be obeyed, additional complexity may also be required of the system design.

C.6 Procedures and organisational measures

Every system has an associated set of procedures of some kind: user procedures (for the operation of the system), maintenance procedures (by the vehicle manufacturer), and service procedures (by a garage). For a safety-related system all these procedures have to be described in detail. The vehicle manufacturer has to consider these procedures during design (or devise the procedures in parallel) in order to get the best coherence between the system and the procedures. Procedures may also be violated by unconscious or subversive actions (e.g. re-programming of engine management chips - see [29]). The performance of a safety analysis is one method to detect possible violations, and hence to correct or diminish the consequences by the addition of extra safety requirements for the design.

During the life-cycle of a safety-critical system there must be an explicit allocation of responsibilities. This means that as safety aspects of the system are allocated to certain parts of the system (e.g. hardware, software, procedures) named persons are made responsible for their development and their integration.

Emphasis must be also be put on standardisation: standards should be understood properly and be applied where possible. The use of standards diminishes the chance of faults occurring, and in general makes it easier for the system to be maintained and used. Procedures must therefore be set up (e.g. ISO 9000) to ensure that all persons involved in development do indeed follow the designated standards.

Consideration must be given to the collection of information about any failures that have occurred, as part of the maintenance procedure. This procedure will itself rely more on the use of computers sited in the garage, and if they are being used to set up vehicle systems that are safety-related, then these maintenance computer systems will themselves be safety-related.

As in-vehicle systems increase in complexity consideration must be given to the training of drivers (of all capabilities).

C.7 Novel systems

The opportunity for developing novel systems provided by the use of programmable devices and communications means that the contents of this document are almost certainly not exhaustive, especially as time goes by. Thus when concepts or components are planned to be used that are not explicitly covered, the basic tenor of this document should be applied to their design, development and use.

C.8 Human behaviour

Some aspects of human behaviour that should be considered during hazard analysis are discussed below.

C.8.1 Human-system interaction

The driver and a vehicle interact in many different ways, and the mechanism for this interaction needs to be considered carefully if it is not to create inadvertently a safety hazard.

C.8.2 Limitations

Humans with normal or at least sufficient vision and hearing base their data gathering on a number of somewhat vague rules, and in addition the data gathered is subject to certain limitations, such as time delay, recognition problems, and of course attentiveness. There is an additional problem that some people with impaired vision or hearing either seem to be unaware of this fact, ignore it, or alternatively they try to live with it. All these are important problems inherent in human data gathering, which require consideration.

There are a number of limitations that have to be especially considered in the case of elderly people, new drivers, and the disabled. Some people, in particular some of the elderly, have limited capabilities for the processing of complex or large quantities of data, and tend to make mistakes where other people would react properly. Some systems might very well aggravate this problem if they are not very carefully designed. New drivers may not have enough experience to recognise dangerous situations and to act accordingly, or may respond in the right manner to an incorrectly perceived danger. Disabled people may have limitations in certain capabilities. All these limitations must be taken into account in the case of safety-related systems, where the limitation might lead to unsafe situations.

C.8.3 Unsafe behaviour

There are some people who appear to enjoy engaging in risky behaviour, and this problem has a tendency to occur in traffic situations; in particular, younger and inexperienced drivers do tend towards over confidence. There is even a vague rule that increased theoretical safety is compensated for by riskier behaviour up to the same level (risk compensation). A developer of a safety-related system should therefore consider unsafe behaviour such as disobedience of the traffic regulations, risky procedures, speeding etc. in an attempt to minimise the risk of misuse.

C.8.4 Unsafe mental state

The use of drugs, medication and alcohol are all a serious threat to the safe use of a vehicle. Sleep shortage, severe fatigue, disturbed rhythms (e.g. jet lag) and other impaired mental states are also threats to safety in general. Although problems in this area can only be handled in a very superficial way, some attention should be paid to them.

C.8.5 Beneficiary problems

Any system is designed for a certain purpose, either with a generic advantage for the users of the system in mind or for a generic policy to be obtained, not necessary for the benefit of the individual "users". In the former case, although in general all users have some advantage from the system, some people might try to turn the safety procedures to their own advantage, and others, on realising this, may behave in an unsafe manner. An example might be a system which ensures that a vehicle keeps a safe distance between it and the vehicle in front. Another vehicle might seek to take advantage of this by endeavouring to slip into that space. This would produce a severe annoyance to the driver trying to keep that safe distance, and he might be tempted to retaliate by manually overriding the automatic system. This kind of human behaviour should be recognised and be addressed in safety requirements.

C.8.6 Human errors

Human errors should be handled in a distinct way and separated from the more general field of concern of human behaviour. In order to understand human errors we should endeavour to determine the internal circumstances and conditions under which humans might go wrong. Errors are more likely in situations of stress, disturbed mental state, lack of experience, lack of knowledge, time pressure, high uncertainty, high cognitive complexity and the sensation of immediate danger, e.g. panic.

Human error probabilities describe the chance of a human error. Human error probabilities should be used during the safety analyses of systems where some human participation can be foreseen, and where humans might cause safety violations. However with a totally new system even expert judgement is difficult because of the lack of operational experience. Table C.1 gives a very approximate estimate of Human Error Probability (HEP) and may be used when no other sources of data are available [19] (see also [20]):

Type of Human Behaviour	HEP
Extraordinary errors: those for which it is difficult to conceive how they could occur: stress free, with powerful cues pointing to success.	10^{-5}
Errors in regularly performed, commonplace simple tasks with minimum stress.	10^{-4}
Errors of commission such as pressing the wrong button or reading the wrong display. Reasonably complex tasks, little time available, some cues necessary.	10^{-3}
Errors of omission where dependence is placed on situation and memory. Complex, unfamiliar task with little feedback and some distraction.	10^{-2}
Highly complex task, considerable stress, little time available.	10^{-1}
Process involving creative thinking, unfamiliar, complex operation where time is short, stress is high.	10^{-1} - 1

Table C.1 - Human Error Probability

C.8.7 Reliance on advice

Any system that issues advice to a driver may not be questioned if it has always been correct in the past. Some drivers may then accept a piece of advice as a command which could be unsafe if the system fails or provides spurious information.

Another manifestation is known as de-skilling which needs to be considered for those systems where the driver is expected to take over in case of a failure. It is possible for drivers to have become so used to having complex decisions being made by the system that, when it fails, they are unable to take over effective manual control, either because they have lost the skill, or they have never had it (e.g. a new driver who has never known anything but the automatic system).

Appendix D — PASSPORT prospective system safety analysis

The following is a summary of the Framework for Prospective System Safety Analysis (PSSA) produced by the EC DRIVE II Project PASSPORT (V2057) [21]. The current version is a draft which will be improved during 1994. Plans are also being made to produce a Computer Assisted Engineering (CAE) tool to support the method.

D.1 Objectives

PSSA is divided into two distinct phases, Preliminary Safety Analysis (PSA) and Detailed Safety Analysis (DSA). A PSA should be carried out before the detailed design is underway and is based on the information that is available at the time. Since most new systems are extensions of, or make use of, existing systems, the information available at even this concept stage can be quite considerable. The objectives of a PSA are:

- preliminary hazard identification.
- preliminary identification of the safety objectives.
- preliminary identification of the safety requirements.
- preliminary assignment of Integrity Levels.

A DSA is performed on the detailed design, and may be repeated as the design changes, or at different levels of detail. The objectives of a DSA are:

- to perform a full hazard analysis.
- to confirm the safety requirements.
- to confirm the assignment of Integrity Levels.
- to confirm the safety measures.

D.2 Preliminary safety analysis

A hazard is an undesirable effect by the system on its environment. It is therefore necessary to produce a model that shows the system in relation to its environment. One such model is the PASSPORT Diagram which is an extension of Yourdon data-flow diagrams. A PASSPORT Diagram (see Figure D.1) contains the system at its centre, labelled "Target of Evaluation", with all the possible interactions with the environment surrounding it. The model is specifically designed for systems which contain computers and which interact with other systems. The diagram can be checked for completeness by ensuring that it will perform the functions specified in the system requirements, and also that all the influences on the system are present (e.g. development process, standards, given data etc.). The diagram can also be checked for consistency by ensuring that "what comes in must go out" and "what comes out must have gone in".

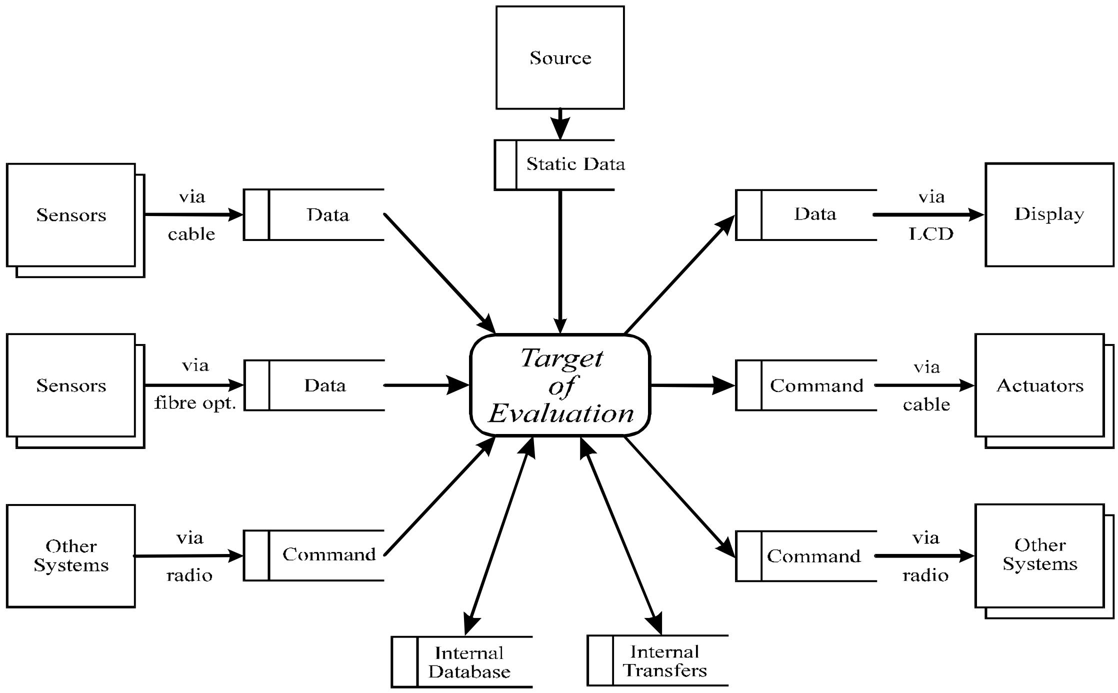

Figure D.1 - PASSPORT Diagram

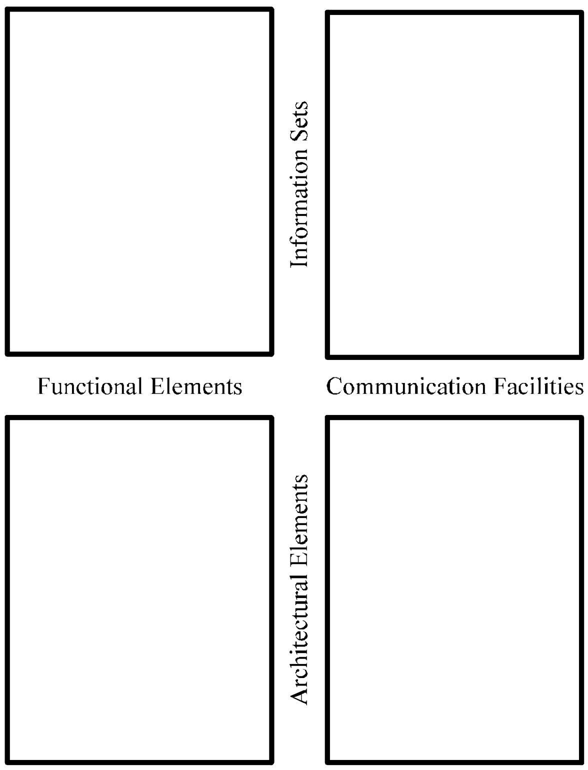

Figure D.2 - PASSPORT Cross

A PSA should be performed by a team of persons with a variety of relevant expertise, including human factors, and once the team is agreed that the PASSPORT Diagram does indeed provide a true representation of the system the hazard analysis can begin. Initially a "what if?" analysis should be performed on each "box" in the diagram; this will enable the primary hazards to be identified, from which the safety objectives can be formulated (see also Appendix C). Each hazard can then be studied further by using a "what causes?" analysis based on the information currently available; this will enable the preliminary safety requirements to be identified. A study of the controllability of each hazard (see Appendix B) will enable a preliminary Integrity Level to be assigned for the design and development of the corresponding sub-system.

D.3 Detailed safety analysis

As the design progresses two models of the system are created, the functional model (functional elements and information sets) and the physical architectural model (architectural elements and communication facilities), often by different teams of engineers (see [28]). The PASSPORT Cross is a modelling technique, based on IBM’s Business Systems Planning, that enables the functional model and the architectural model to be brought together and checked for consistency. Currently the PASSPORT Cross (see Figure D.2) consists of four matrices, although other matrices may prove to be useful. The top left and bottom right connection matrices define the functional model and the architectural model respectively. The top right and bottom left projection matrices provide the means of relating the functional model to the architectural model. The PASSPORT Cross can be checked for consistency by ensuring that each entry in a matrix has at least one entry in every other matrix (the current complete set of consistency checks can be found in [21]). Whilst it is possible to produce a PASSPORT Cross at any level of detail required, a CAE tool is really necessary to handle the vast quantities of information involved.

Once the PASSPORT Cross has been shown to be consistent it can be used to confirm the results of the PSA, as well as to ensure that the design has not introduced any new hazards. The PASSPORT Cross can also be used to readily identify critical architectural elements, i.e. those used by many functional elements, whether safety-related or not. A Failure Mode and Effects Analysis (FMEA) [9, 10] can be performed with the assistance of the PASSPORT Cross, and then a Fault Tree Analysis (FTA) [12, 13] can be done on each hazard found, also with the assistance of the PASSPORT Cross. The number of levels of decomposition for which a detailed safety analysis should be performed will depend on the Integrity Level.

D.4 Traceability

By using a suitable labelling scheme, e.g. PASSPORT Identification for Traceability System (PITS), it is possible to keep track of the functional model and the architectural model as they are decomposed, in particular of those functions that implement the safety requirements. By this means it is always possible to readily identify all those parts of the system that are safety-related and subject them to any special analysis that may be required.

Appendix E — Configuration and measures of fault detection

If safety cannot be obtained sufficiently with quality control (measures for fault avoidance) during the development and production phase, then measures must be taken to control errors during the operation phase.

In general there is a basic safety requirement for the higher levels of integrity that a single fault should never lead to a dangerous situation. To obtain this goal errors in the state of the system must be detected and an appropriate protection measure must be carried out. The second requirement is that, if an error is undetected, it should never lead to a dangerous situation even in combination with one or more other faults. These safety requirements can be mainly fulfilled with the aid of two methods. The first uses the comparison of the results of redundant systems. The second uses self-test procedures to detect errors in the system.
Initially a system architecture must be specified, this is related to the level of integrity and the possible process conditions (whether there be a safe state or not, permitted failure tolerance time, etc.) in the event of an error. Secondly, adequate test methods to recognise errors must be selected. Thirdly, the organisational requirements (preventive maintenance) to detect faults later in the life-cycle of the system must be determined.

Some appropriate structures, fault detection and organisational measures are described in the following sections. Other adequate measures, which have at least the same effect on safety could also be applied [3, 22, 23].

E.1 Processes with a safe state and without a safe state

We have to distinguish between processes with a safe state and those without a safe state. In the first case the process can be transferred into the safe state after the detection of dangerous errors. In the second case fault tolerant structures are required to perform the operation continuously after fault detection (fail operational). Fault tolerance can be achieved by redundancy. In general, redundancy means the provision of additional elements or systems so that any one can perform the required function regardless of the state of operation or failure of any other. A further possibility is to continue operation with degraded functional capabilities or performance in case of failures (fail soft, limp home or graceful degradation).

E.2 Protection and control systems

A safety-related system can be used as a protection system or can be a part of a control system. Protection systems receive several inputs from the monitored process and their sole purpose is to transfer the process into a safe state if they recognise a transition to a potentially dangerous state of that process. Control systems use information from the process to influence process operation. In this case we have to distinguish between processes with a safe state and those without a safe state. If the safety-related system is part of the control system then it is

often more economical, particularly in complex systems, to separate the safety tasks from the main tasks.

E.3 System structure

The safety-related system can be structured as a single-channel or a multi-channel (programmable) electronic system.

E.3.1 Single-channel structures

Single-channel structures consist of a functional unit which is used to execute specified operations.

The system can be tested:

- by an initial functional test.
- by an internal self-test.
- by a self-test and additional monitoring units.

Advantage

- Simple hardware construction.

Disadvantage

- Not all random faults can be detected.
- Extensive test routines may need a long time to execute.
- It is not always possible to recognise faults in time.
- There is only one way to switch into a safe state.

Objective

To fulfil the specified function as far as possible without a large amount of hardware redundancy.

Faults to be covered

Most random hardware faults.

E.3.1.1 Single-channel with function test

The single-channel structures are tested to see if their functions are correct before they begin operation. Once the single-channel structure is in operation, no more tests are carried out, and the risk of a dangerous failure is considered acceptable.

The function test consists essentially of one or more operational values which are typical for the task to be processed. These operational values are given to the single-channel structure and the reactions of the single-channel structure are compared with reference reactions according to the specification. If the reactions coincide, the single-channel structure is assumed to be fault free - continuity between the test points is assumed. The function test however, only detects errors which depend on the information flow. More complex errors in the state of the system are generally not detected.

E.3.1.2 Single-channel with self-test

The self-tests are used to detect latent faults which are information flow independent in sub-units (RAM, ROM, CPU etc.). Once a fault is detected the system should be switched into a safe state and an error message should be issued (see Figure E.1).

E.3.1.3 Single-channel with self-test and monitoring

The system includes self-tests of sub-units and internal monitoring functions which are used to monitor the correctness of data, the compliance of marginal values, the run time control of the program etc. Additional hardware and software is needed, e.g. a watchdog.
Once a fault is detected the system should be switched into a safe state and an error message should be issued.

E.3.1.4 Single-channel with self-test, monitoring and independent protective sub-system

As distinct from the single-channel structures described above an additional independent protective sub-system is used to carry out the safety actions. An error message should be issued.

E.3.2 Multi-channel structures

Multi-channel structures consist of independent redundant functional units without any mutual hazardous influence on each other. In order to fulfil the requirements of the higher Integrity Levels, multi-channel structures should be considered.

Objective

If a fault occurs, either the operation must be maintained or a safety action must be carried out by another part of the system. The former can be realised with a fault tolerant structure and the latter with a fail-safe design.

Faults to be covered

- Random hardware faults.
- Systematic faults (only with diverse redundant channels).
- Malfunction caused by external influences (Environmental, EMI).

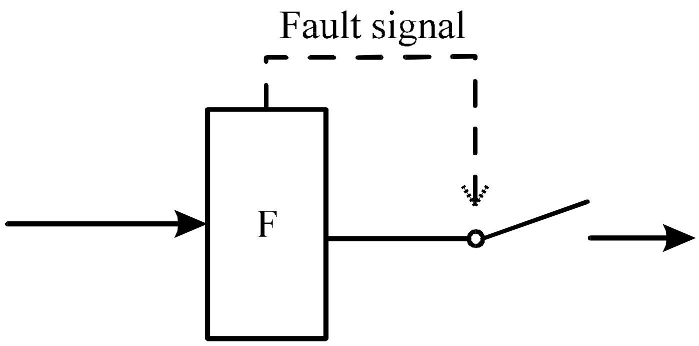

Figure E.1 - Single Channel with Self Test

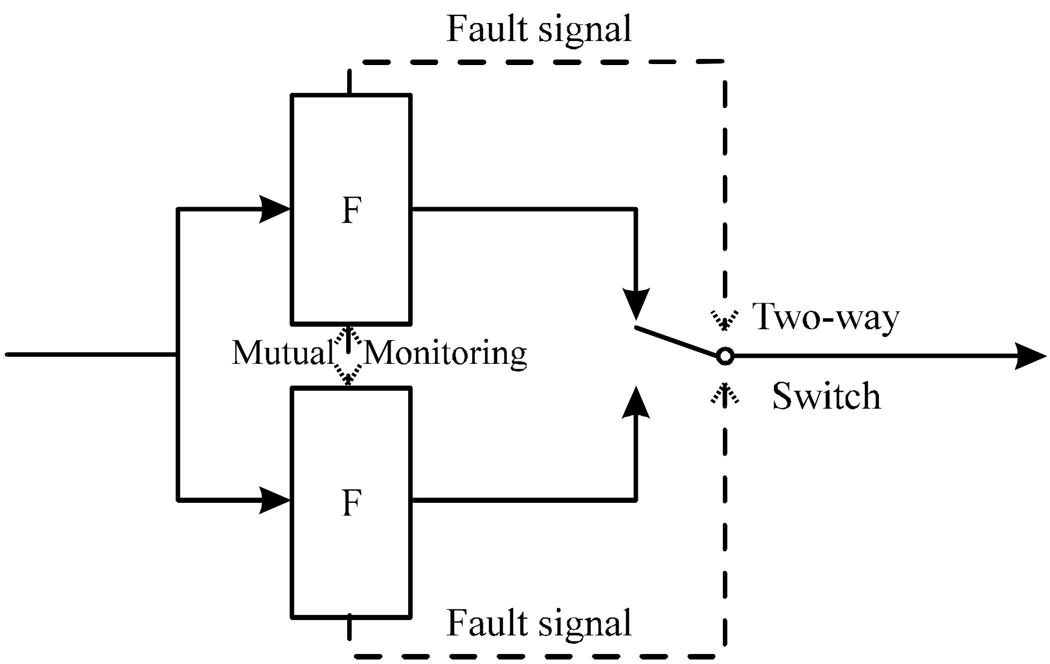

Figure E.2 - Two Channel Structure without a Safe State

Redundant structures

Redundant channels can be built using homogeneous or diverse techniques.

If systematic faults cannot be reduced sufficiently then the channels should be designed diversely. Diversity can be attained by several approaches. For example: microprocessors from more than one source, or one channel is programmable whilst another is non-programmable (with possible reduced functionality) (e.g. electronic, mechanic, pneumatic, hydraulic). Diverse programming techniques are discussed in [31]. The degree of diversity necessary will depend on the Integrity Level of the system.

Systematic failures in multi-channel systems caused by the same influences are called "common mode failures" or "common cause failures". Even if the system is structured with a high degree of diversity, the safety requirements specification itself must nevertheless be considered as a potential source of a common mode failure.

Reporting

Whenever an error is detected and the structure is automatically re-configured it is essential that the user is made aware of this fact so that the problem can be rectified as soon as possible. This is particularly important for fail-operational systems where no degradation of performance will be noticed.

E.3.2.1 Two-channel structures

E.3.2.1.1 Two-channel structures without a safe state

The operation of the system should be maintained if a fault occurs in one of the channels (see Figure E.2).

A 2-way switch selects which channel controls the process. Errors must be detected in each channel. If a fault occurs in the active channel the switch over to the other channel must be carried out automatically. The 2-way switch and the error recognition mechanisms must be guaranteed against breakdowns in a suitable manner, for example by using a fail-safe technique.

During safety analysis, a fault tolerant two-channel structure with a 2-way switch is comparable to a single-channel structure. A fault tolerant structure is necessary where no safe state is available and the function has to be maintained.

The following methods are suitable to detect faults in the individual channels:

a) The use of self-tests
Self-tests such as those in single-channel systems are used to detect faults.
b) The use of self-tests and monitoring
Self-tests and monitoring functions similar to those in single-channel systems are used

to recognise errors. The tests should be diverse to avoid common mode failures.

c) Mutual monitoring

Both channels are used for mutual monitoring. In this case the problem is how to decide which channel is defective; this can be solved by also carrying out independent self-tests in both channels.

E.3.2.1.2 Two-channel structure with a safe state

The system should be transferred into a safe state if a fault occurs (see Figure E.3). Two-channel structures contain independent and feedback-free functional units to execute the specified functions independently in each case. To recognise errors the system includes a unit which compares internal signals (e.g. bus) and/or input/output signals. Both channels can be designed homogeneously or diversely. At least two switch-off paths should exist so that the channels themselves, as well as the comparator can cause a switch-off.

The comparison can take place in various ways (hardware or software). In the case of diverse double channels, only the comparison of output signals and intermediate results is generally possible. The comparison of internal signals (e.g. bus comparisons) requires homogeneous channels and extensive synchronisation. The comparator itself also has to be subject to fault detection. This procedure recognises all errors which are detected as different values during the comparison. Supplementary measures have to be taken for errors which are information flow independent and which are not capable of being detected by comparison.

These supplementary measures may be:

- regular additional activation and testing of the hardware and software components
- regular self-tests of the components concerned
- additional monitoring of the components concerned

E.3.2.2 Three-channel structures

There are several possible structures applicable such as 1 out of 3, 2 out of 3 or 3 out of 3. Only the 2 out of 3 structures are considered in the following discussion, because they are the most common. These structures can be used both for fault tolerant systems and for systems with a safe state (see Figure E.4).

E.3.2.2.1 Three-channel structure (2 out of 3) without a safe state

The operation of the system must be maintained without interruption. The majority evaluation is carried out with a fault-free voter. If a failure occurs in one channel then the function will be maintained as a two-channel structure without a safe state (see Section E.3.2.1.1). This structure can therefore be designed to cater for two successive faults and still remain operational.

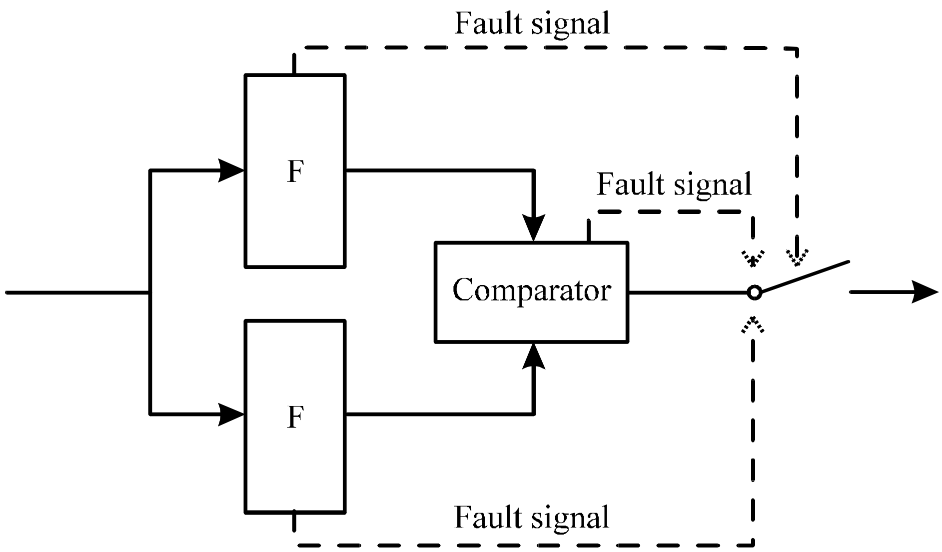

Figure E.3 - Two-Channel Structure with a Safe State

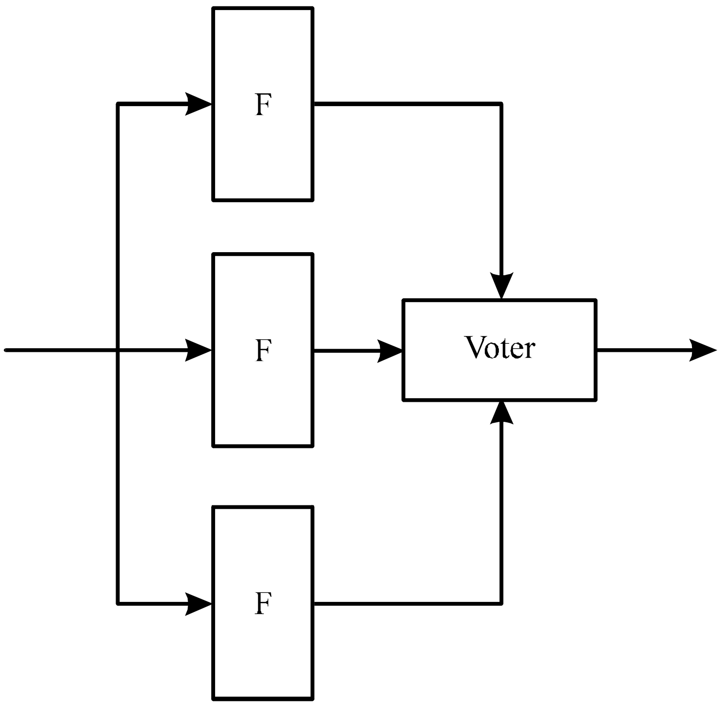

Figure E.4 - Three-Channel Structure

E.3.2.2.2 Three-channel structure (2 out of 3) with a safe state

These structures are used in applications which have a safe state, and if high requirements are additionally made on their availability.

The system must be designed in such a manner that the voter/comparator recognises an error in a channel and switches the system over to a two-channel structure with safe state (see Section E.3.2.1.2).

E.4 Fault detection measures

In addition to the structural measures described above, measures are also required to detect faults in components such as the CPU, memory, sensors, actuators etc. The effectiveness of these measures can generally be ranged from average to great. The measures are divided into technical and non-technical and a possible set is listed below.

Technical Measures

CPU

- Standard CPU Test
- CPU Test with great effectiveness

Monitoring Units

- Watchdog
- Comparator
- Voter
- Other Units

ROM

- Parity Bit Technique
- Multi-Bit Redundancy Technique
- Longitudinal Redundancy Check (LRC), Simple Check Word
- Cyclic Redundancy Check (CRC), Simple Check Word
- Cyclic Redundancy Check (CRC), Multiple Check Word
- Double ROM with hardware or software comparison

RAM

- RAM Test Checkerboard
- RAM Test March
- RAM Test Walkpat (Walking Pattern)
  The memory field which is to be tested is first uniformly loaded. The first cell

is then inverted and the remaining area of the memory is tested against a correct background. Finally, the first cell is read and tested again, inverted once again i.e. written as it was originally, and the whole process is repeated with the next and following cells. A second run with the "walking bit pattern" (hence the name) is carried out with an inverse background loaded [24].

- RAM Test Galpat (Galloping Pattern)
  This technique is similar to the Walkpat test but involves a more rigorous test reading procedure [25].
- RAM Test Transparent Galpat
- Parity Bit Technique
- Multi-Bit Redundancy Technique
- Double RAM with hardware or software comparison

Input and Output Data

- Parity Bit Technique
- Cyclic Test of Input and Output Units
- Code Check of Input and Output Data
- Redundancy Check
- Multi-Channel Parallel Output
- Rewrite of Outputs
- Comparison of Inputs
- Fault Detection through Monitoring of the Technical Process
- Plausibility Check

Internal Data Transfer

- One-Bit Hardware Redundancy
- Multi-Bit Hardware Redundancy
- Double BUS Structure
- Cyclic Test of Data Paths
- Time Redundancy
- Information Redundancy

Supply Units

- Power Supply Monitoring
- Clock Monitoring
- Cooling System (Ventilation) Monitoring

Other Devices

- Sensors test
- Actuators test

Software

- Plausibility Check
- Time Monitoring of Program Tasks
- Watchdog
- Logical Monitoring of Program Sequences
- Combination of Time and Logical Monitoring of Program Sequences
- Diversity of Software

Non-Technical Measures

Periodical Inspection, Preventive Maintenance

- Effect Test
- Component Test

Organisational Measures

- Qualified Personnel for Maintenance
- Technical Manual
- User Manual
- Modification
- Plausibility Test of Operator Input
- Others

Appendix F — Quality assurance plan

This appendix describes a possible quality assurance plan based upon ISO 9000-3 (TickIT) [16].

F.1 Quality system - Framework

This should be a high level document which should identify the software quality control functions and activities, and assign specific responsibilities and authority to ensure that they are carried out. The quality system (framework) part of the plan should normally contain the following main headings:

F.1.1 Management responsibility

The responsibility, authority and the interrelation of all personnel who manage, perform and verify work affecting quality shall be defined. In particular the assessor should be satisfied that the software quality control manager has the authority to act independently and to resolve problems. If part or all of the software is sub-contracted then procedures will be needed to define the obligations on the vehicle manufacturer to ensure, by inspection, auditing and review, that the sub-contractor is following the same software quality control procedures as those used in the rest of the project. The assessor has to be given access to the sub-contractor (see [32]).

F.1.2 Quality system

A quality system should be established and maintained to provide an integrated process throughout the entire life-cycle.

There should be plans for periodic and systematic reviews of the software quality control system.

F.1.3 Internal quality system audits

Regular quality system audits should be carried out to verify whether the quality activities comply with planned arrangements, and to determine the effectiveness of the quality system. The results should be documented and corrective action be taken if appropriate.

F.1.4 Corrective action

It is inevitable that mistakes will be made and problems will arise during the software development; it is to be hoped that the software quality control procedures will be sufficiently robust and rigorous to identify all of these faults and problems before Job 1. There is therefore a requirement for a plan to set out procedures to handle the reporting and analysis

of problems, and for procedures for the re-testing of the software. It is particularly important to periodically review the effectiveness of these procedures.

Plans should be made for the procedures to be followed for the acceptance of software that for some reason cannot be made to conform with the original specifications (see Section F.2.8). This will involve the assessor, the quality control manager as well as the project manager.

F.2 Quality system - Life-cycle activities

This is the start of the technical level and plans are needed here particularly to help identify features or aspects of the project which are new and unusual, i.e. features which will require special care and attention if they are to be implemented successfully. The plans should stipulate how these novel or unfamiliar aspects will be handled, e.g. the safety considerations of the product or system. The quality related activities should be planned and implemented with respect to the nature of the life-cycle model chosen (see Appendix H). The quality system (life-cycle activities) part of the plan should, normally, contain the following headings:

F.2.1 Contract review

This subject is concerned with the situation when there is a formal relationship between the supplier of the software and the vehicle manufacturer. Procedures should be established and maintained for contract review and the handling of exceptions and changes (see [32]). Where there is no formal contractual relationship (e.g. the manufacturer is using an in-house team) it is still good practice to hold a "start-up" meeting to ensure that all parties have the same understanding of what is required when, who is responsible for each part of the task, and that suitable resources are available

F.2.2 Purchaser’s requirements specification

The requirements specification plan should identify and justify the use of the techniques and tools that will be used for the functional requirements specification, and the safety integrity requirements specification. Different techniques are appropriate for different levels of integrity (see Appendix G). The requirements should be stated precisely enough so as to allow validation during product acceptance. There should be methods for agreeing requirements and approving changes.

F.2.3 Development planning

The development plan identifies the various phases of the development, the respective inputs and outputs, and the plans for the verification of each phase. Different techniques are appropriate for different levels of integrity (see Appendix G). The plan should cover the organisation of the resources of the project and its management. Progress reviews should also be planned.

F.2.4 Quality planning

The quality plan should address the quality objectives and how they will be addressed.

F.2.5 Safety planning

The safety plan should address the safety objectives and how they will be addressed.

F.2.6 Design and implementation

Because of the complexity of software products, it is imperative that these activities are carried out in a disciplined manner. There should be a wide ranging set of plans for documentation standards and coding conventions and practices.

An analysis should be performed on the plans for the design to judge the degree of success necessary for each phase; these requirements must be consistent with the acceptance criteria (Section F.2.8)

There should also be a section on the use of specific CASE tools. It should identify each software engineering tool, technique and methodology to be used, and should explain how the use of these CASE tools and procedures will contribute towards the software quality control. It should also document the validation and/or reliability of the tools.

Different techniques are appropriate for different levels of integrity (see Appendix G).

F.2.7 Testing and validation

The test plan will have links into the verification and validation plans and links from the design plan. The tests must have defined objectives and show that the components of the system work together and that no requirements have been ignored (see [31]). The results of the tests must be recorded and any problems that are discovered that may have an effect on another part of the system must be tracked until they are resolved.

There should be systematic and documented verification and validation reviews. These reviews take place throughout the life-cycle and should cover management, requirements capture, design, implementation and testing. The plans should identify the objectives of each individual review activity and the composition of the review team. The verification and validation plan should refer to the corrective action plan when problems have been identified by the reviews. This subject is covered in more detail in [31].

The range and rigour of the test and verification and validation procedures is strongly related to the need to meet the requirements of any given Integrity Level (see Appendix G).

F.2.8 Acceptance

The acceptance criteria should be stated, and the method of handling problems detected during

the acceptance procedure should be agreed and documented.

F.2.9 Replication, delivery and installation

This should cover issues such as copyright, custody of master and back-up copies, and the period of obligation of the supplier to provide copies.

F.2.10 Maintenance

The maintenance activities must be planned very carefully, both to maintain the Integrity Level of the software after any changes, and also because of the unusually long lifetime of in-vehicle software, when access to the necessary information could be difficult.

F.3 Quality system - Supporting activities

The quality system (supporting activities) part of the plan covers those processes that need to be maintained throughout the life-cycle. It should normally contain the following headings:

F.3.1 Configuration management

The configuration management plan has to include the identification, control, accounting and audit of all software items. The rigour of the configuration management process varies with Integrity Level (see Appendix G).

F.3.2 Document control

The document control plan should cover the determination of those documents which should be subject to the control procedures, and the procedures themselves (see [31]).

F.3.3 Quality records

Procedures should be established and maintained for the identification, collection, indexing, filing, storage, maintenance and disposal of quality records.

F.3.4 Measurement

Metrics can be applied to both the development process and to the software itself (see [30])

F.3.5 Rules, practices and conventions

These are the rules, practices and conventions to make the quality system effective. They should be reviewed and revised as required.

F.3.6 Tools and techniques

These are the tools and techniques used to make the quality system effective. They should be reviewed and improved as required.

F.3.7 Purchasing

There should be a plan to ensure that a purchased product or service conforms to specified requirements.

F.3.8 Included software support

Procedures should be established and maintained for the validation, storage, protection and maintenance of any existing software product that is not subject to the main life-cycle (e.g. software re-use).

F.3.9 Training

Procedures should be established and maintained to identify the training needs, and provide for the training of all personnel performing activities affecting quality.

Appendix G — Assessment requirements

The following sections describe the method of obtaining a level of confidence in a software product that corresponds to the Integrity Level identified during hazard analysis. The contents of this Appendix have been summarised in Table 2, with Integrity Level 0, not safety-related, not fully included.

The assessment of software that is being re-used can impose its own difficulties since whilst the necessary information may not be available, it may, however, bring with it a good previous track record; this issue is discussed further in [31].

G.1 Specification and design

Integrity Level 0

- The minimum requirement is a natural language description or specification of the high level functions of the software, together with a description of the software architecture. A structured approach should be taken.

Integrity Level 1

- A structured method should be used.

Integrity Level 2

- A structured method, supported by a CASE tool, should be used.

Integrity Level 3

- A formal specification should be made for the functions identified to be at this level, with a natural language commentary.

Integrity Level 4

- A formal specification of the complete software system should be made with a natural language commentary. The code should be automatically produced (refined) directly from this specification through a series of mathematically correct stages (when available).

G.2 Languages and compilers

Contrary to the impression that is sometimes given in language manuals most, if not all, languages do not have precisely defined semantics. This means that not only may compilers contain faults, but different compilers for the same language may implement a given feature in different ways. In addition some programming "features" such as pointers or recursion can cause unpredictable behaviour in certain circumstances. This subject is discussed in more detail in [26].

The objective of assessment is to ensure that the final code does reflect the specification to the required degree of confidence. For this reason it is normally recommended that strongly typed and structured languages (e.g. ADA or Pascal) should be used, which provide compiler and run-time system assistance in the finding of faults, rather than the "flexible" languages (e.g. C or Assembler) which require additional checks to be made by other means. Whilst it is extremely undesirable for such checks to be made by hand, when validation tools are used the resulting development system might be simpler than a large complex compiler. Formal arguments of correctness are easier to apply to a high-level language program than to one written in an assembly language. However, since there are no formally proven compilers yet available, it may be necessary to show that the machine code does indeed reflect the high-level language version of the program (see Section G.5). For this reason, or for reasons of the required speed of execution, restricted memory availability etc. assembly languages are sometimes still being used at all levels of integrity. When assembly languages are used this must normally be justified, particularly at high Integrity Levels; also the programs should be kept small with a structure imposed upon them, and static analysis performed.

Integrity Level 0

- Whilst there are no restrictions for this non-safety-related level, it is recommended that consideration should be given as to whether, through changing requirements (adaptive maintenance), the software may be required to increase its Integrity Level during its lifetime.

Integrity Level 1

- A structured language should be used that conforms to an accepted national or international standard.

Integrity Levels 2–3

- A restricted subset of a structured language with automatic syntax checking, type checking and range checking should be used. The restricted subset should avoid potentially insecure language constructs e.g. pointers or recursion.
- Some versions of some compilers are put through a formal validation process by an approved assessor; tests are also done by BSI on other compilers, and the results are published. Only validated or tested compilers should be used (if available), which provide traceability of the source code to the object code to demonstrate determinism.

Integrity Level 4

- Only independently certified compilers with a formally defined syntax and semantics, and proof that the compiler meets them, should be used (when they become available, until then only validated or tested compilers should be used).

G.3 Configuration management

This is a wide ranging process and it can be subdivided to help simplify the requirements at each level. It has to be noted that, though subdivided, all topics within this process relate to each other.

The configuration management system should:

a) uniquely identify the versions of each software item;
b) identify the versions of each software item which together constitute a specific version of a complete product;
c) identify the build status of software products in development or delivered and installed;
d) control simultaneous updating of a given software item by more than one person;
e) provide co-ordination for the updating of multiple products in one or more locations as required;
f) identify and track all actions and changes resulting from a change request, from initiation through to release.

G.3.1 Version and configuration control - products

Procedures should be established and maintained for identifying software items during all life-cycle phases, starting from specification through development, replication, delivery and maintenance. Each individual software item should have a unique identification. The point at which version control takes place must be stated in the quality assurance plan, and traceability is provided by the change and build control process (see Section G.3.2). Version and configuration control should be applied to:

Integrity Level 0
- The product as a whole.

Integrity Level 1
- The modules, documents and manuals.
- The source code.

Integrity Levels 2–4
- All relationships, e.g. hierarchical, parent-child, between all software products under version control.
- All the tools used in the development process.

G.3.2 Change and build control - processes

The following processes should be set up:

Integrity Levels 0
- Unique identification of those items covered by the configuration management

system (see Section G.3.1).

Integrity Level 1

- A change control system that will ensure that the product matches the documentation.
- An access control system that avoids simultaneous updating of a software item by more than one person.
- A change control system that will ensure only authorised changes are permitted. A different system may be required for baseline changes from that used during normal development.

Integrity Level 2

- A system to control and audit changes.
- A means of confirming that the software has been built from identifiable components. This process will vary depending on whether it is performed by the customer or the supplier.

Integrity Levels 3–4

- An automated process should be used for both change control and build control.
- An automated check should be performed to confirm that only intended changes are made, and to identify modules affected by a change in any other module.

G.4 Testing

Although the following indicates the degree of testing required for a particular Integrity Level, for commercial or legal reasons it may be necessary to do more, especially at the lower levels. Since full and complete testing is usually not feasible, thought must be given to the completion criteria necessary for the product. This must be planned in conjunction with the verification and validation requirements, in particular the test coverage criteria used (e.g. statement, branch or path) must be justified. The requirements are specified either in terms of techniques to be used, or the objective to be achieved.

Integrity Level 0

- A test plan should be created.
- The product should be shown to be fit for its purpose.

Integrity Level 1

- Requirements testing against all safety requirements.
- Tests should be repeatable and traceable.

Integrity Level 2

- Black box testing against an appropriate level of architectural description.

Integrity Level 3

- White box testing of modules to a defined degree of coverage.
- Stress testing against livelock and deadlock.
- Syntactic static analysis.

Integrity Level 4

In the following, if 100% testing is not possible then all omissions must be both fully justified, and shown to make up the complete set of possible tests.

- 100% white box structural coverage for each module.
- 100% requirements testing.
- 100% integration testing (data connectivity).
- Semantic static analysis.

G.5 Verification and validation

The topics mentioned in this section are discussed in detail in [31].

Integrity Levels 0–1

- Show tests to be suitable.
- Show tests have been performed.
- Show test results to be acceptable.
- Show tests exercise safety features.
- Correction of errors should be traceable.

Integrity Level 2

- Structured program review.
- Show that no new problems are introduced by correcting faults found by testing (regression testing).

Integrity Level 3

- Automated static analysis of formally specified code.
- Formal proof or proof argument of safety properties in formally specified code.
- Analysis for lack of livelock and deadlock.
- Document test coverage and justify why the gaps have not been tested.
- Review tests of specified safety features and show all interactions between safety features have been covered.
- Show that tests confirm the validity of the formal analysis of formal specifications.

Integrity Level 4

- All software tools being used should have been formally validated to this Integrity Level (when they become available)
- Formal proof or proof argument of formally specified code against its specification.
- Formal proof or proof argument for lack of livelock and deadlock.

- Show that the object code does reflect the source code.

G.6 Access for assessment

Assessment can only proceed if the assessor has the right of access, subject to appropriate confidentiality agreements, to the relevant information and the ability to judge the validity of that information. This is of particular importance when a sub-contractor is used ([32]).

Integrity Level 0

- Visual inspection of requirements and acceptance criteria.

Integrity Level 1

- Access to quality assurance and product related plans.
- Inspection of the company training policy.
- Access to system test results

Integrity Level 2

- Access to design documentation and software test results.
- Inspection of company training structure.

Integrity Level 3

- Inspection of the techniques, processes and tools to be used, and the conformance of the work to them.
- Testing in the presence of a witness.
- Evidence of adequate training of the development personnel on the tools and techniques.
- Access to code.

Integrity Level 4

- Full access to all stages and processes.

Appendix H — Plans and life-cycles

A documented plan is essential for the production of a system if it is to be traceable, testable, maintainable and assessable. The plan will lay down the project management and organisation policy, standards of design, programming practices, quality control and maintenance procedures. An assessment is necessary to judge whether the plans are effective and appropriate for the task, and whether they have been followed during the life-cycle. The overall plan will incorporate detailed plans for the organisation of specific activities based on a life-cycle model of development. The first stage in this planning process is the selection of the life-cycle model itself.

There are many models for the system development cycle and it is necessary to choose one that covers the needs of the organisation and any standards that it is following (e.g. [3, 16, 27]). Whilst no life-cycle model can reflect every facet of a development, it is necessary to impose a structure on the development to prevent the uncontrolled chaos that it can too easily slide into, especially with software. The combination of a life-cycle, an agreed understanding of how to interpret it, and a disciplined adherence to it, forms the foundations for the evaluation and assessment processes. The system life-cycle upon which this document is based is shown in Figure H.1.

Figure H.1 is a version of the waterfall model which emphasises the sequential nature of development in that one process must finish, normally with a formal review, before the succeeding one can begin. Another version of the waterfall model is the V model shown in Figure H.2 which emphasises the verification and validation processes throughout the development.

It is very seldom that development takes place in the rigid sequential manner that might be implied by Figures H.1 and H.2. In order to emphasise the concurrent nature of many of the development processes the structure of the life-cycle of high quality software can also be pictured as being made up of a sequence of software development processes and a parallel set of project management processes, as shown in Figure H.3. The number of sequential stages and the number of parallel activities vary with the choice of the model; e.g. some split the design stage into architectural design and detailed design. Feedback of information within a project is extremely useful and desirable. Good feedback mechanisms using regular reviews, with adequate and easy to use documentation systems, allow the results from project checks to fine tune the outcome of a project. In severe cases feedback of information to the correct department or people can prevent major inconsistencies from developing further.

The choice and interpretation of the structure of a life-cycle model depends to a large extent on the purpose for which it is being used. In the Interim DEFSTAN 00-55 [27] the MOD states that each stage of the development process must be completed before the next is started, and so this may be required as part of a contract. However in DO178B [15] the regulated initiation of any sequential stage is permitted provided sufficient information is available from the previous stage. It also provides mechanisms for feed-back from down-stream stages to correct errors of commission or omission detected in up-stream stages. The criteria for the

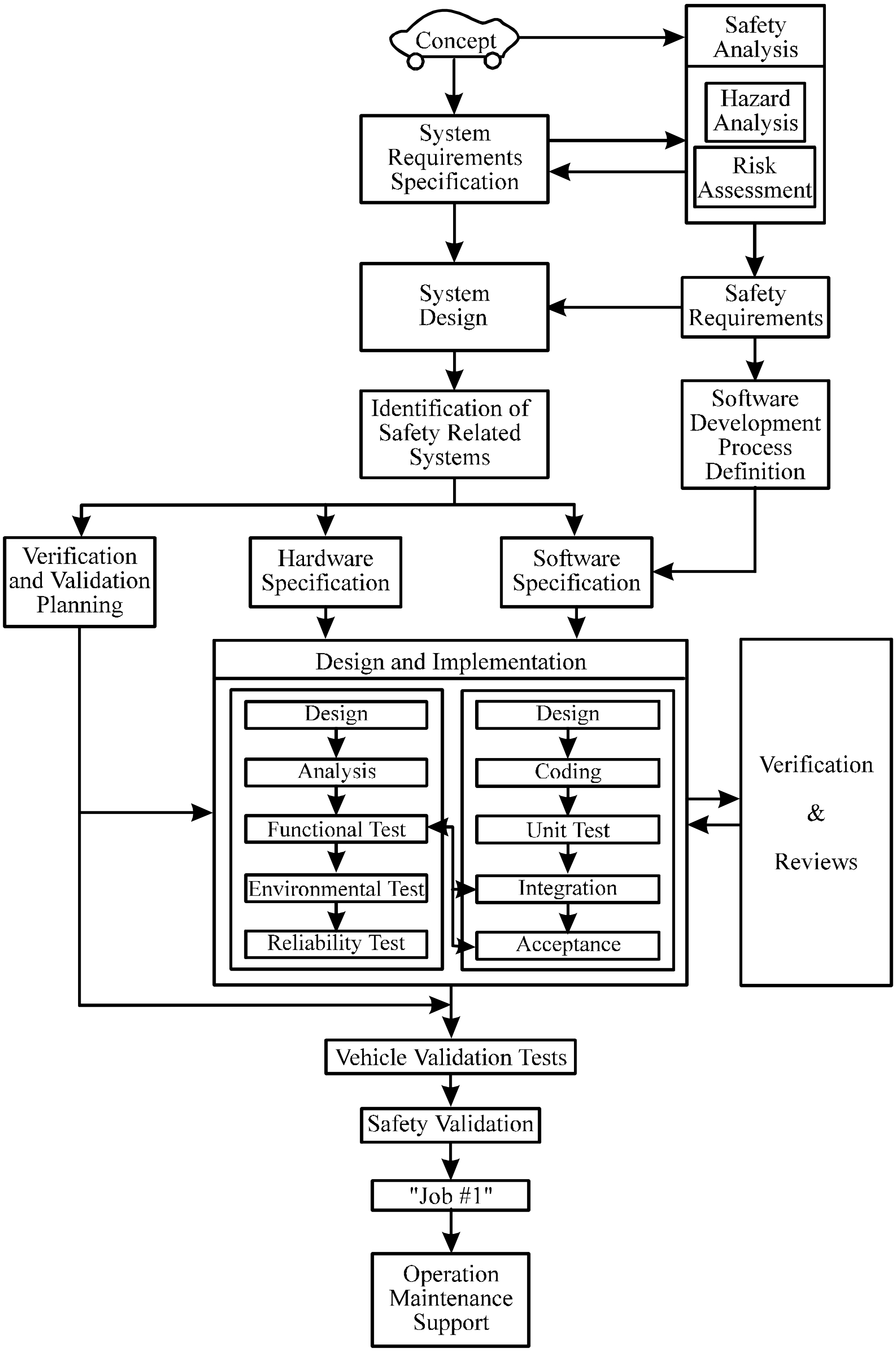

Figure H.1 - System Life-cycle incorporating Software and Safety Life-cycles

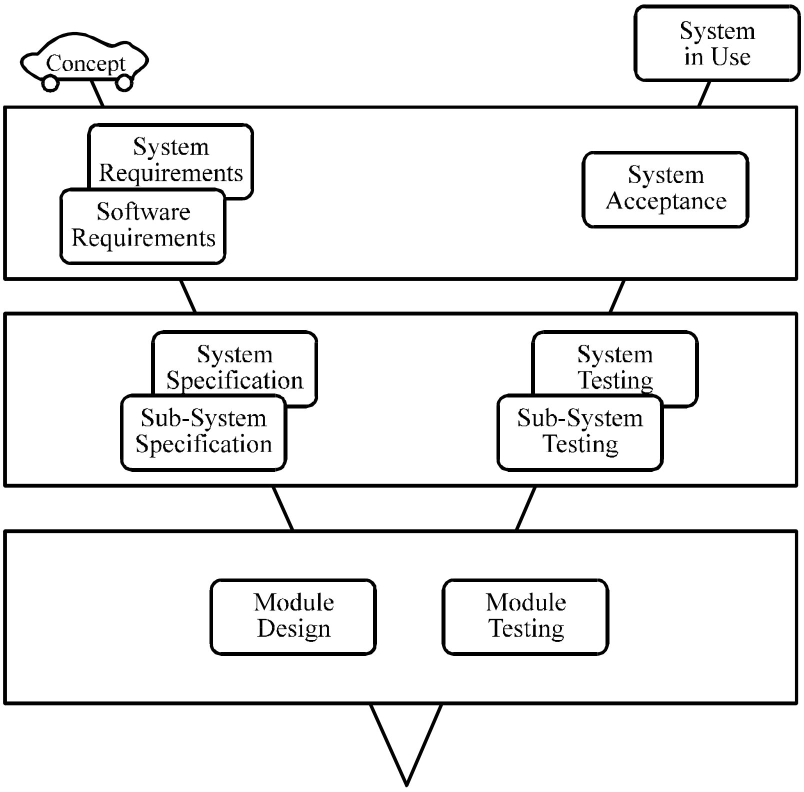

Figure H.2 - V Life-cycle Model

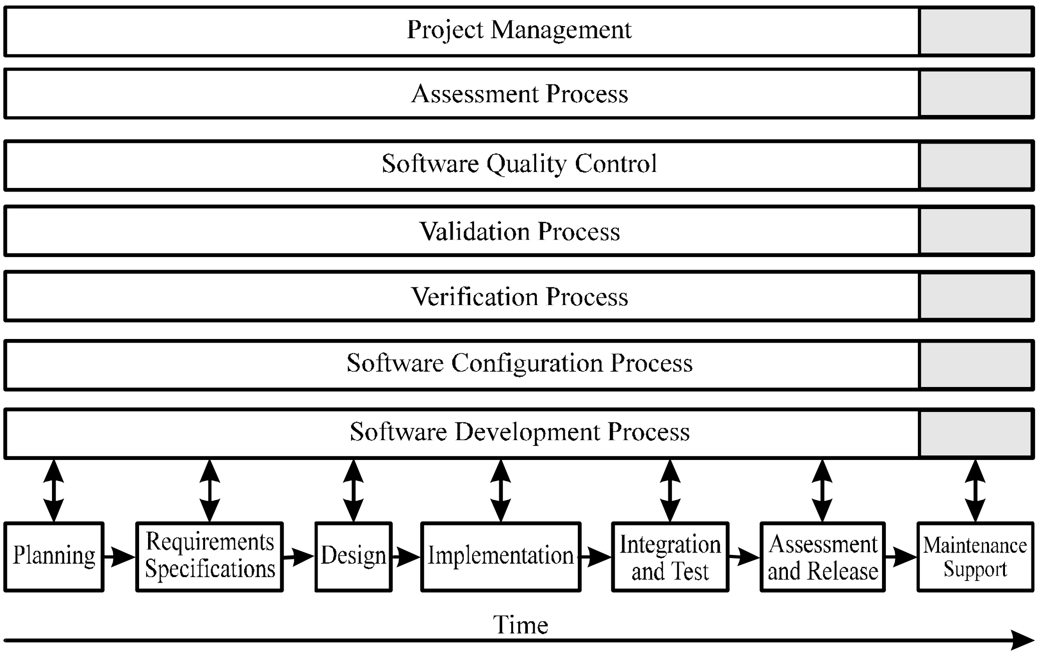

Figure H.3 - Software Life-cycle

accepted completion of any particular stage are set jointly by the assessment process, the software quality control process and the verification process. A life-cycle should normally be incorporated into a quality assurance plan.

References

[1] Collins, *The Collins Paperback English Dictionary*, Collins 1986.

[2] HSE, *Programmable Electronic Systems in Safety Related Applications*, HMSO, 1987.

[3] IEC SC65A WG10 (Technical Committee No. 65: Industrial — Process Measurement and Control), *Draft — Functional Safety of Electrical/Electronic/Programmable Electronic Systems: Generic Aspects*, International Electrotechnical Commission, 1992.

[4] DEFSTAN 00-56, *Safety Management Requirements for Defence Systems Containing Programmable Electronics (Interim)*, MOD, 1993

[5] DIN V19250, *Fundamental Safety Aspects to be Considered for Measurement and Control Equipment*, DIN, Germany, 1988.

[6] DRIVE Safely, *Towards a European Standard: The Development of Safe Road Transport Informatic Systems (Draft 2)*, EC DRIVE I Project V1051, 1992.

[7] Imai M and Kaizen, *The Key to Japan's Competitive Success*, McGraw Hill, 1986.

[8] Automotive Electronics Reliability Handbook, SAE, Vol. AE9, ISBN 0-89883-009-5, 1987.

[9] IEC 812, *Analysis Techniques for System Reliability - Procedure for Failure Mode and Effects Analysis (FMEA)*, International Electrotechnical Commission, 1985.

[10] SMMT, *Guidelines to Failure Mode and Effects Analysis*, Society of Motor Manufacturers and Traders, 1989.

[11] ITSEC, *Information Technology Security Evaluation Criteria*, Commission of the European Communities, ISBN 92-826-3004-8, 1991.

[12] IEC 1025, *Fault Tree Analysis (FTA)*, International Electrotechnical Commission, 1990.

[13] NUREG-0492, *Fault Tree Handbook*, U. S. Nuclear Regulatory Commission, 1986.

[14] STARTS Public Purchaser Group, *The STARTS Purchasers' Handbook: Procuring Software-Based Systems*, NCC Publication, 1989.

[15] DO178B, *Software Consideration in Airborne Systems and Equipment Certification, (Interim Draft 6.3)*, RCTA, 1992.

[16] TickIT, *A Guide to Software Quality Management System Construction and*

Certification using EN29001, Department of Trade and Industry, 1992.

[17] Limnios N and Jeanette J P, *Event Trees and their Treatment on PC Computers*, Reliability Engineering, Vol. 18, No. 3, 1987.

[18] Nielsen D S, *The Cause Consequence Diagram Method as a Basis for Quantative Accident Analysis*, RISO-M-1374, 1971.

[19] Comer P J and Kirwan B J, *A Reliability Study of a Platform Blowdown System*, Automation for Safety in shipping and Offshore Petroleum Operations, Elsevier, ISBN 0-444-7010-1, 1986.

[20] Kirwan B, *Human Reliability Assessment*, in "Evaluation of Human Work", Ed. Wilson J and Corlett E N, Taylor and Francis, London, 1990.

[21] PASSPORT, *Framework for Prospective System Safety Analysis*, EC DRIVE II Projects V2057/8, 1994.

[22] DIN VDE V 0801, *Principles for Computers in Safety Related Systems*, DIN, Germany, 1989.

[23] DEFSTAN 00-41 (Parts 1-5), *Practices and Procedures for Reliability and Maintainability*, Ministry of Defence, UK, 1981-1989.

[24] Barraclough W, Chiang A and Sohl W, *Techniques for testing the microcomputer family*, Proceedings of the IEEE, Vol. 64, No 6, pp 943-950, 1976.

[25] Henk de Jonge J and Smulders A J, *Moving Inversions Test Patterns is Thorough, yet Speedy*, Computer Design, No 5, 1976.

[26] Cullyer W J, Goodenough S J and Wichmann B A, *The Choice of Computer Language for Use in Safety-Critical Systems*, Software Engineering Journal, Vol. 6, No 2, March 1991.

[27] DEFSTAN 00-55, *The Procurement of Safety-Critical Software in Defence Equipment (Interim)*, MOD, 1991.

[28] *Diagnostics and Integrated Vehicle Systems*, MISRA Report 1, 1994.

[29] *Software in Control Systems*, MISRA Report 4, 1994.

[30] *Software Metrics*, MISRA Report 5, 1994.

[31] *Verification and Validation*, MISRA Report 6, 1994.

[32] *Subcontracting of Automotive Software*, MISRA Report 7, 1994.

Bibliography

IEC SC65A WG9, *Software for Computers in the Application of Industrial Safety-Related Systems*, International Electrotechnical Commission, 1991.

Rook P, *Software Development Process Models*, in Software Reliability Handbook, Ed. P Rook, Elsevier Applied Science, 1990.

Somerville I, *Software Engineering (4th Edition)*, Addison Wesley, 1992.